User manual

myDatalogEASY IoT GW ATEX

Valid from:

- **Firmware version: 01v031**
- **Server version: 50v007**
- **Hardware version: 1.1**



301176 | Rev.05

Microtronics

Chapter 1 Table of contents

Cover ... **1**
Chapter 1 Table of contents ... **3**
Chapter 2 Declaration of conformity ... **13**
2.1 myDatalogEASY IoT GW ATEX 2G/4G EU ... 13
2.2 myDatalogEASY IoT GW ATEX 2G/3G/4G World ... 14
2.3 myDatalogEASY IoT GW ATEX 2G/M1/NB1 World ... 15
Chapter 3 Ex certification ... **17**
Chapter 4 Specifications ... **19**
Chapter 5 General specifications ... **23**
5.1 Translation ... 23
5.2 Copyright ... 23
5.3 General descriptive names ... 23
5.4 Ex protection ... 24
5.5 Safety instructions ... 25
5.5.1 Use of the hazard warnings ... 25
5.5.2 General safety instructions ... 26
5.5.3 Safety and preventative measures for handling GSM/GPRS modems ... 26
5.5.3.1 Safety and precautionary measures for the GSM/GPRS modem installation ... 27
5.5.3.2 Safety measures for installing the antenna ... 27
5.6 Overview ... 28
5.6.1 Block diagram ... 29
5.7 Intended use ... 30
5.8 General product information ... 30
5.9 Device labelling ... 32
5.10 Installation of spare and wear parts ... 33
5.11 Storage of the product ... 33
5.12 Warranty ... 33
5.13 Disclaimer ... 34
5.14 Obligation of the operator ... 34
5.15 Personnel requirements ... 35
Chapter 6 Functional principle ... **37**

6.1 Recommended procedure ... 39
6.1.1 Development of M2M/IoT application ... 39
6.2 Functionality of the internal data memory ... 39
6.3 Memory organisation ... 41
6.4 Procedure in case of connection aborts ... 42
6.4.1 Connection abort in "online" mode ... 43
6.4.2 Connection abort during a Device Logic download ... 43
6.5 Timeout monitoring in online mode ... 43
6.6 Automatic selection of the GSM network ... 43
6.7 Determining the GSM/UMTS/LTE signal strength ... 44
6.8 Determining the GSM position data ... 44
6.9 Error handling ... 44
6.10 Registration memory blocks ... 45
6.10.1 REG_APP_OTP ... 45
6.11 File transfer ... 46
6.12 Meaning of the SIM state ... 47
6.13 Using the external SIM slot ... 48
Chapter 7 Storage, delivery and transport ... 51
7.1 Inspection of incoming deliveries ... 51
7.2 Scope of supply ... 51
7.3 Storage ... 52
7.4 Transport ... 52
7.4.1 Transporting power supply units ... 52
7.5 Return ... 53
Chapter 8 Installation ... 55
8.1 Dimensions ... 55
8.2 Inserting/replacing the SIM card ... 56
8.3 Sealing the pressure compensation ... 58
8.4 Installing the myDatalogEASY IoT GW ATEX ... 61
8.4.1 Wall mounting ... 62
8.4.2 Pipe mounting ... 63
8.4.3 Outdoor installation ... 64
8.4.3.1 Attaching the housing for outdoor installation to a wall ... 64

8.4.3.2 Attaching the housing for outdoor installation to a pipe ... 65
8.5 Safety instructions for cabling ... 66
8.5.1 Information on preventing electrostatic discharges (ESD) ... 67
8.6 Electrical installation ... 67
8.6.1 Supply concepts ... 67
8.6.1.1 Powered by primary battery ... 68
8.6.1.2 Powered by PV module ... 68
8.6.1.3 Powered by mains adapter ... 69
8.6.2 Connection of the power supply ... 71
8.6.3 Connecting the mobile network antennas ... 74
8.6.4 Technical details about the USB interface ... 75
8.6.5 Technical details about the Bluetooth Low Energy interface ... 75
8.6.6 Technical details about energy management ... 75
8.6.7 Technical details about the energy supply ... 76
8.6.7.1 PSU713 BP (300526) ... 76
8.6.7.2 PSU DC (300529) ... 77
8.6.8 Technical details about the system time ... 77
Chapter 9 Initial Start-Up ... 79
9.1 User information ... 79
9.2 Applicable documents ... 79
9.3 General principles ... 79
9.4 Commissioning the system ... 79
9.5 Testing communication with the device ... 81
Chapter 10 User interfaces ... 83
10.1 User interface on the myDatalogEASY IoT GW ATEX ... 83
10.1.1 Operating elements ... 83
10.1.1.1 Solenoid switch ... 83
10.1.1.2 Three-colour LED ... 84
10.1.1.3 Lid reed contact ... 84
10.2 User interface on the myDatanet server ... 84
10.2.1 Site configuration ... 84
10.2.1.1 Site ... 84
10.2.1.2 Comments ... 85

10.2.1.3 Control ... 85
10.2.1.4 Configuration 0 - Configuration 9 ... 85
10.2.1.5 Alarm settings ... 86
10.2.1.6 Basic settings ... 87
10.2.1.7 FTP export settings ... 87
10.2.2 Device configuration ... 88
10.2.2.1 Comments ... 88
10.2.2.2 Measurement instrument ... 88
10.2.2.3 GPRS ... 89
Chapter 11 DeviceConfig ... 91
11.1 General ... 91
11.2 Prerequisites ... 91
11.3 Functional principle ... 92
11.3.1 USB BLE-Adapter ... 93
11.4 Installation ... 93
11.4.1 Installing USB BLE-Adapter driver ... 95
11.5 Menu of the DeviceConfig ... 96
11.5.1 Settings ... 96
11.5.1.1 Options ... 96
11.6 Connecting a Device via USB ... 97
11.7 Connecting a Device via Bluetooth Low Energy ... 98
11.8 "GSM" tab ... 99
11.9 "Log" tab ... 101
11.10 "Firmware" tab ... 103
11.11 "Features" tab ... 104
11.12 "Sync" tab ... 104
11.12.1 Existing connection to the myDatalogEASY IoT GW ATEX ... 105
11.12.2 No connection to a device ... 106
11.13 Recommended procedure ... 107
11.13.1 Synchronisation with the myDatanet server ... 107
11.13.1.1 Internet connection available when reading out the data ... 107
11.13.1.2 No Internet connection when reading out the data ... 111
Chapter 12 "tbd" smartphone app ... 115

12.1 General ... 115
Chapter 13 myDatanet server ... 117
13.1 Overview ... 117
13.1.1 Explanation of the symbols ... 117
13.2 "Customer" area ... 118
13.3 "Sites / Applications" area at customer level ... 120
13.3.1 Reports ... 121
13.3.2 Map view ... 121
13.4 Recommended procedure ... 121
13.4.1 Creating the site ... 121
Chapter 14 rapidM2M Studio ... 125
14.1 General ... 125
14.2 Prerequisites ... 126
14.3 Project dashboard ... 127
14.4 CODEbed ... 128
14.5 TESTbed ... 129
Chapter 15 Device Logic ... 131
15.1 General ... 131
15.1.1 Direct entry of a device logic ... 131
15.1.2 Uploading a binary file ... 131
15.1.3 Using the CODEbed of the web-based development environment rapidM2M Studio ... 131
15.2 Device API ... 132
15.2.1 Constants ... 132
15.2.2 Timer, date & time ... 133
15.2.2.1 Arrays with symbolic indices ... 133
15.2.2.2 Constants ... 133
15.2.2.3 Functions ... 133
15.2.3 Uplink ... 138
15.2.3.1 Arrays with symbolic indices ... 138
15.2.3.2 Constants ... 139
15.2.3.3 Callback functions ... 143
15.2.3.4 Functions ... 144
15.2.4 System ... 154

15.2.4.1 Arrays with symbolic indices ... 154
15.2.4.2 Functions ... 154
15.2.5 Encoding ... 155
15.2.5.1 Constants ... 155
15.2.5.2 Functions ... 156
15.2.6 Bluetooth Low Energy ... 161
15.2.6.1 Arrays with symbolic indices ... 161
15.2.6.2 Constants ... 163
15.2.6.3 Callback functions ... 166
15.2.6.4 Functions ... 168
15.2.7 Registry ... 175
15.2.7.1 Constants ... 175
15.2.7.2 Callback functions ... 176
15.2.7.3 Functions ... 176
15.2.8 Position ... 180
15.2.8.1 Arrays with symbolic indices ... 180
15.2.8.2 Constants ... 182
15.2.8.3 Functions ... 183
15.2.9 Math ... 189
15.2.10 Char & String ... 192
15.2.11 CRC & hash ... 200
15.2.11.1 Arrays with symbolic indices ... 200
15.2.11.2 Functions ... 200
15.2.12 Various ... 201
15.2.12.1 Arrays with symbolic indices ... 201
15.2.12.2 Constants ... 202
15.2.12.3 Functions ... 202
15.2.13 Console ... 208
15.2.14 SMS ... 210
15.2.14.1 Callback functions ... 210
15.2.14.2 Functions ... 210
15.2.15 External SIM ... 211
15.2.15.1 Arrays with symbolic indices ... 211

15.2.15.2 Functions ... 211
15.2.16 File transfer ... 212
15.2.16.1 Arrays with symbolic indices ... 212
15.2.16.2 Constants ... 212
15.2.16.3 Callback functions ... 212
15.2.16.4 Functions ... 214
15.2.17 LED ... 218
15.2.17.1 Constants ... 218
15.2.17.2 Functions ... 219
15.2.18 Solenoid switch ... 221
15.2.18.1 Constants ... 221
15.2.18.2 Callback Funktionen ... 221
15.2.18.3 Functions ... 222
15.2.19 Power management ... 223
15.2.19.1 Arrays with symbolic indices ... 223
15.2.19.2 Constants ... 224
15.2.19.3 Callback functions ... 224
15.2.19.4 Functions ... 225
15.3 Device Logic error codes ... 226
15.4 Syntax ... 230
15.4.1 General syntax ... 230
15.4.1.1 Format ... 230
15.4.1.2 Optional semicolons ... 230
15.4.1.3 Comments ... 230
15.4.1.4 Identifier ... 230
15.4.1.5 Reserved keywords ... 230
15.4.1.6 Numerical constants ... 231
15.4.1.6.1 Numerical integer constants ... 231
15.4.1.6.2 Numerical floating-point constants ... 231
15.4.2 Variables ... 231
15.4.2.1 Declaration ... 231
15.4.2.2 Local declaration ... 231
15.4.2.3 Global declaration ... 231

15.4.2.4 Static local declaration ... 232
15.4.2.5 Static global declaration ... 232
15.4.2.6 Floating point values ... 232
15.4.3 Constant variables ... 232
15.4.4 Array variables ... 232
15.4.4.1 One-dimensional arrays ... 232
15.4.4.2 Initialisation ... 233
15.4.4.3 Progressive initialisation for arrays ... 233
15.4.4.4 Multi-dimensional arrays ... 233
15.4.4.5 Arrays and the "sizeof" operator ... 234
15.4.5 Operators and expressions ... 235
15.4.5.1 Notational conventions ... 235
15.4.5.2 Expressions ... 235
15.4.5.3 Arithmetic ... 235
15.4.5.4 Bit manipulation ... 236
15.4.5.5 Assignment ... 236
15.4.5.6 Comparative operators ... 237
15.4.5.7 Boolean ... 237
15.4.5.8 Other ... 238
15.4.5.9 Priority of the operators ... 238
15.4.6 Statements ... 239
15.4.6.1 Statement label ... 239
15.4.6.2 Composite statements ... 240
15.4.6.3 Expression statement ... 240
15.4.6.4 Empty statement ... 240
15.4.6.5 Assert expression ... 240
15.4.6.6 Break ... 241
15.4.6.7 Continue ... 241
15.4.6.8 Do statement while (expression) ... 242
15.4.6.9 Exit expression ... 242
15.4.6.10 For (expression 1; expression 2; expression 3) statement ... 242
15.4.6.11 Goto label ... 243
15.4.6.12 If (expression) statement 1 else statement 2 ... 243

15.4.6.13 Return expression ... 243
15.4.6.14 switch (expression) {case list} ... 243
15.4.6.15 While (expression) statement ... 244
15.4.7 Functions ... 245
15.4.7.1 Function arguments ("call-by-value" versus "call-by-reference") ... 245
15.4.7.2 Named parameters versus fixed parameters ... 247
15.4.7.3 Standard values of function arguments ... 247
15.5 Differences to C ... 248
Chapter 16 Data Descriptor ... 251
16.1 Data structure ... 251
16.1.1 Division of a structured measurement data channel into individual data fields ... 252
16.1.2 Division of a configuration memory block into individual data fields ... 253
16.1.3 Division of the aloha data into individual data fields ... 254
16.1.4 Attributes of the field definition ... 254
16.2 Example ... 259
16.3 Special values of the data types ... 261
Chapter 17 API ... 263
17.1 Backend API ... 263
17.2 rapidM2M Playground ... 263
17.2.1 Overview ... 264
Chapter 18 Maintenance ... 265
18.1 General maintenance ... 265
18.2 Replacing the power supply unit ... 266
Chapter 19 Removal/disposal ... 269
Chapter 20 Troubleshooting and repair ... 271
20.1 General problems ... 271
20.2 Log entries and error codes ... 272
20.2.1 Modem error ... 277
20.3 Evaluating the device log ... 279
20.3.1 Evaluating the device log on the myDatanet server ... 279
20.3.2 Evaluating the device log using DeviceConfig ... 279
Chapter 21 Spare parts and accessories ... 281
21.1 Chargeable features ... 281

21.2 Assembly sets ... 281
21.3 Antennas ... 281
21.4 Power supply units ... 281
21.5 Charging devices and power supply units ... 282
21.6 Solar panel ... 282
21.7 Cable ... 282
21.8 BLE sensors ... 282
21.9 BLE output modules ... 282
21.10 Other accessories ... 282
Chapter 22 Document history ... 283
Chapter 23 Glossary ... 291
Chapter 24 Contact information ... 293

Chapter 2 Declaration of conformity

2.1 myDatalogEASY IoT GW ATEX 2G/4G EU

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung: Product: Désignation du produit:	Frei programmierbares Gatway zur Erfassung, Verarbeitung und Übertragung von Daten		
Type : Type code: Type:	**myDatalogEASY IoT GW ATEX 2G/4G EU**	**Gültig ab:** Valid from: Valide à partir de:	**Rev. 1.0**

Microtronics

Hersteller:
Manufacturer :
Fabricant:

Microtronics Engineering GmbH
Hauptstrasse 7
A-3244 Ruprechtshofen

Das bezeichnete Produkt stimmt mit den folgenden Europäischen Richtlinien überein. Eine oder mehrere der in der zugehörigen Baumusterprüfbescheinigung SIQ 22 ATEX 388 X genannten Normen wurden durch neue Ausgaben ersetzt. Der Hersteller erklärt für das vorstehend genannte Produkt auch die Übereinstimmung mit den Anforderungen der neuen Normenausgaben."
The designated product is in conformity with the following european directives. One or more of the associated listed in the Type Examination Certificate SIQ 22 ATEX 388 X standards have been replaced by new editions. The manufacturer for the above product also compliance with the requirements of the new standard editions.
Le produit décrit est conforme aux directives européennes suivantes. Une ou plusieurs des normes énumérées dans l'attestation de type concernant SIQ 22 ATEX 388 X étaient replacées par des nouvelles éditions. Le producteur déclare pour le produit décrit ci-dessus la conformité aux exigences des nouvelles éditions des normes.

		Europäische Norm	
(2014/30/EU)	**EMC Directive**		
		EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62368-1+A11:2017 EN62311 EN62479	
	EMC 3.1b	EN301489-1 V2.1.1 EN301489-52 V1.1.0 EN301489-17 V3.1.1	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-13 V13.1.1 EN300328 V2.2.2 EN300330 V2.1.1	
(2014/34/EU)	**ATEX Directive**		
		EN IEC 60079-0 EN 60079-7	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	
	Kennzeichnung/ Markings/ Marquage CE Ex II 3G Ex ec nC IIC T4 Gc		

Ruprechtshofen, den 19.06.2023

Ort und Datum der Ausstellung Place and date of issue Lieu et date d'établissement	Andreas Zuser , Ex-Schutzbeauftragter Unterschrift name and signature of authorised person Nom et signature de la personne autorisée	Hans-Peter Buber, Managing Director Unterschrift name and signature of authorised person Nom et signature de la personne autorisée

2.2 myDatalogEASY IoT GW ATEX 2G/3G/4G World

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung: Product: Désignation du produit:	Frei programmierbares Gatway zur Erfassung, Verarbeitung und Übertragung von Daten		
Type : Type code: Type:	**myDatalogEASY IoT GW ATEX 2G/3G/4G World**	**Gültig ab:** Valid from: Valide à partir de:	**Rev. 1.0**

Microtronics

Hersteller: Manufacturer : Fabricant:	**Microtronics Engineering GmbH** Hauptstrasse 7 A-3244 Ruprechtshofen

Das bezeichnete Produkt stimmt mit den folgenden Europäischen Richtlinien überein. Eine oder mehrere der in der zugehörigen Baumusterprüfbescheinigung SIQ 22 ATEX 388 X genannten Normen wurden durch neue Ausgaben ersetzt. Der Hersteller erklärt für das vorstehend genannte Produkt auch die Übereinstimmung mit den Anforderungen der neuen Normenausgaben."

The designated product is in conformity with the following european directives. One or more of the associated listed in the Type Examination Certificate SIQ 22 ATEX 388 X standards have been replaced by new editions. The manufacturer for the above product also compliance with the requirements of the new standard editions.

Le produit décrit est conforme aux directives européennes suivantes. Une ou plusieurs des normes énumérées dans l'attestation de type concernant SIQ 22 ATEX 388 X étaient replacées par des nouvelles éditions. Le producteur déclare pour le produit décrit ci-dessus la conformité aux exigences des nouvelles éditions des normes.

		Europäische Norm	
(2014/30/EU)	**EMC Directive**		
		EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62368-1+A11:2017 EN62311 EN62479	
	EMC 3.1b	EN301489-17 V3.1.1 EN301489-1 V2.2.3 EN301489-52 V1.2.0	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-2 V13.1.1 EN301908-13 V13.1.1 EN300328 V2.2.2 EN300330 V2.1.1	
(2014/34/EU)	**ATEX Directive**		
		EN IEC 60079-0 EN 60079-7	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	
	Kennzeichnung/ Markings/ Marquage CE Ex II 3G Ex ec nC IIC T4 Gc		

Ruprechtshofen, den 19.06.2023

Ort und Datum der Ausstellung Place and date of issue Lieu et date d'établissement	Andreas Zuser , Ex-Schutzbeauftragter Unterschrift name and signature of authorised person Nom et signature de la personne autorisée	Hans-Peter Buber, Managing Director Unterschrift name and signature of authorised person Nom et signature de la personne autorisée

2.3 myDatalogEASY IoT GW ATEX 2G/M1/NB1 World

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung: Product: Désignation du produit:	Frei programmierbares Gatway zur Erfassung, Verarbeitung und Übertragung von Daten		
Type : Type code: Type:	**myDatalogEASY IoT GW** **ATEX2G/M1/NB1 World**	**Gültig ab:** Valid from: Valide à partir de:	**Rev. 1.0**

Microtronics

Hersteller:
Manufacturer :
Fabricant:

Microtronics Engineering GmbH
Hauptstrasse 7
A-3244 Ruprechtshofen

Das bezeichnete Produkt stimmt mit den folgenden Europäischen Richtlinien überein. Eine oder mehrere der in der zugehörigen Baumusterprüfbescheinigung SIQ 22 ATEX 388 X genannten Normen wurden durch neue Ausgaben ersetzt. Der Hersteller erklärt für das vorstehend genannte Produkt auch die Übereinstimmung mit den Anforderungen der neuen Normenausgaben."
The designated product is in conformity with the following european directives. One or more of the associated listed in the Type Examination Certificate SIQ 22 ATEX 388 X standards have been replaced by new editions. The manufacturer for the above product also compliance with the requirements of the new standard editions.
Le produit décrit est conforme aux directives européennes suivantes. Une ou plusieurs des normes énumérées dans l'attestation de type concernant SIQ 22 ATEX 388 X étaient replacées par des nouvelles éditions. Le producteur déclare pour le produit décrit ci-dessus la conformité aux exigences des nouvelles éditions des normes.

		Europäische Norm	
(2014/30/EU)	**EMC Directive**		
		EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62368-1+A11:2017 EN62311 EN62479	
	EMC 3.1b	EN301489-1 V2.1.1 EN301489-52 V1.1.0 EN301489-17 V3.1.1	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-13 V13.1.1 EN300328 V2.2.2 EN300330 V2.1.1	
(2014/34/EU)	**ATEX Directive**		
		EN IEC 60079-0 EN 60079-7	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	
	Kennzeichnung/ Markings/ Marquage CE Ex II 3G Ex ec nC IIC T4 Gc		

Ruprechtshofen, den 19.06.2023

Ort und Datum der Ausstellung Place and date of issue Lieu et date d'établissement	Andreas Zuser , Ex-Schutzbeauftragter Unterschrift name and signature of authorised person Nom et signature de la personne autorisée	Hans-Peter Buber, Managing Director Unterschrift name and signature of authorised person Nom et signature de la personne autorisée

Chapter 3 Ex certification

(1) **TYPE EXAMINATION CERTIFICATE**

(2) Product Intended for use in Potentially Explosive Atmospheres – **Directive 2014/34/EU**

(3) Type Examination Certificate Number:

SIQ 22 ATEX 388 X **Issue: 0**

(4) Product: Datalogger, type: myDatalogEASY Iot GW ATEX xx

(5) Manufacturer: Microtronics Engineering GmbH

(6) Address: Hauptstrasse 7, 3244 Ruprechtshofen, Austria

(7) This product and any acceptable variation thereto are specified in the schedule to this certificate and the documents therein referred to.

(8) SIQ Ljubljana certifies that this product has been found to comply with the Essential Health and Safety Requirements relating to the design and construction of products intended for use in potentially explosive atmospheres, given in Annex II to the Directive 2014/34/EU.

The examination and test results are recorded in the confidential test report TEx184/22.

(9) Compliance with the Essential Health and Safety Requirements has been assured by compliance with:

EN IEC 60079-0:2018 + AC:2020 **EN IEC 60079-7:2015 + A1:2018**

EN IEC 60079-15:2019

Where additional criteria beyond those given here have been used, they are listed at item (18) in the schedule to this certificate.

(10) If the sign "X" is placed after the certificate number, it indicates that the product is subject to Specific Conditions of Use specified in the schedule to this certificate.

(11) This Type Examination Certificate relates only to the design and construction of the specified product in accordance with the Directive 2014/34/EU. Further requirements of the Directive apply to the manufacturing process and supply of this product. These are not covered by this certificate.

(12) The marking of the product shall include the following:

Ex II 3 G Ex ec nC IIC T4 Gc

Certification body Ljubljana, 25 November 2022

Bojan Pečavar

Digitally signed by BOJAN PEČAVAR
DN: cn=BOJAN PEČAVAR gn=BOJAN c=SI o=SIQ LJUBLJANA
Date: 2022-11-25 12:45+01:00

Page 1/5

The Type Examination Certificate is valid only if signed. The certificate may be reproduced only in full and without changes. Any extracts and changes shall be approved by SIQ Ljubljana.

SIQ Ljubljana, Mašera-Spasićeva ulica 10, SI-1000 Ljubljana, +386 1 4778 221, ex@siq.si

(13) **SCHEDULE**

(14) **Type Examination Certificate Number SIQ 22 ATEX 388 X, Issue: 0**

(15) Description of Product

Datalogger, type: myDatalogEASY Iot GW ATEX xx, is a universal datalogger for receiving data from external sensors via Bluetooth communication and sending the collected data to base station via wireless connection. The datalogger is intended for fixed installation and it can be supplied by external DC voltage supply via connector or by internal battery pack with primary batteries.

Datalogger is designed in type of protection increased safety "ec", except the reed relays are hermetically sealed which corresponds type of protection "nC".

Technical data

- Type key: myDatalogEASY Iot GW ATEX xx,

where "xx" can be variable parameter depending on type of communication network, i.e. 3G World, 2G/M1/NB1 World, 2G/4G EU or 2G/4G World.

- Power supply:
  - PSU DC: External DC supply, from 12 V d.c. to 32 V d.c.
  - PSU713BP: Internal primary battery pack, 7.2 V, 13 Ah
- Ambient temperature range: From -20°C to +50°C.

(16) Test Report

TEx184/22 dated 25 November 2022.

(17) Specific Conditions of Use

- Ambient temperature range -20°C ≤ T_{amb} ≤ +50°C shall be considered.
- Only power supply units PSU DC or PSU713BP provided by manufacturer shall be used.
- Datalogger shall be installed in area with low risk of mechanical danger.
- The device shall be protected against direct sunlight or other sources of UV radiation.
- Potential electrostatic charging hazard: Clean the device with moist or electrostatically dissipative cloth.
- Connector for external power supply shall always be connected to external circuit or protected by intended plug.
- Opening of the enclosure and connection or disconnection of the external circuit are allowed only when explosive atmosphere cannot be present.
- Metal guard for connectors shall be earthed.

(18) Essential Health and Safety Requirements

Compliance with the Essential Health and Safety Requirements has been assured by compliance with the requirements of the standards listed under item (9).

The Type Examination Certificate is valid only if signed. The certificate may be reproduced only in full and without changes. Any extracts and changes shall be approved by SIQ Ljubljana.

SIQ Ljubljana, Mašera-Spasićeva ulica 10, SI-1000 Ljubljana, +386 1 4778 221, ex@siq.si

Chapter 4 Specifications

Voltage supply	Battery: • PSU713 BP : 13Ah Direct power supply: • PSU DC : 2kV overvoltage protection and reverse voltage protection Additional information is provided in "Technical details about the energy supply" on page 76.
Supply or charging voltage	12...32VDC (max. 12W)
Housing	Material: ABS / PC (housing/cover) Weight: 770g (without power supply unit) Degree of protection: IP66 / IP68 (Connections plugged/unplugged) IP68: immersion depth max.1m for min. 105 days Dimensions (WHD): 130 x 240 x 84mm (without antenna)
Ex certification	 II 3G Ex ec nC IIC T4 Gc
Operating temperature	-20...+50°C
Air humidity	15...90%rH non-condensing
Storage temperature	-20...+50°C
Display	Three-colour LED with selectable function: • Signalling the operating state (controlled by the FW) • Function freely useable (controlled by the device logic)
Operation	Solenoid switch with selectable function: • Initiation of the transmission (evaluated by the FW) • Function freely useable (evaluated by the device logic) Lid reed contact: • Function freely useable (evaluated by the device logic)
Antenna connector [1)]	up to 2 x FME-M
System time	Hardware real-time clock with its own buffer battery (service life >10 years) and automatic time synchronisation with the server Additional information is provided in "Technical details about the system time" on page 77.

Internal sensors	Internal housing temperature • Measurement range: -20...+60°C • Resolution: 0,1°C Humidity in the housing • Measurement range: 0...100% rH • Resolution: 0,1% rH
Bluetooth	5.0 compatible low energy module
USB interface	1 x mini-B USB 2.0 slave for the connection to a PC. The DeviceConfig configuration program must be installed on the PC or the web-based development environment rapidM2M Studio must be used to enable communication with the myDatalogEASY IoT GW ATEX . Additional information is provided in "Technical details about the USB interface" on page 75.
Data memory	3MB internal flash memory. The size of the data records is variable (max. 1023 Byte) and is determined by the device logic created by the user. The system-related overhead is 11 Byte per data record. Additional information is provided in"Functionality of the internal data memory" on page 39.
Configuration memory	10 independent blocks each with 4000 Bytes
Registration memory	Flash: 4 blocks each with 1kB and pre-defined purposes for storing device-specific data RAM: 1 optional block with max. 1kB for storing application-specific data Additional information is provided in "Registration memory blocks" on page 45.
Memory for the binary	256kB (uncompressed size) Additional information is provided in "Memory organisation" on page 41.
Data transmission	Bluetooth Low Energy: Range: 20m (depending on the environmental conditions) 2G/4G Europe (myDatalogEASY IoT GW ATEX 2G/4G EU) • 2G GPRS 900MHz / 1800MHz • LTE CAT1 B3, B7, B20 2G/3G/4G world (myDatalogEASY IoT GW ATEX 2G/3G/4G World) • 2G GPRS 900MHz / 1800MHz • 2G GPRS 850MHz / 1900MHz • UMTS B1, B2, B5, B8 • LTE FDD B1, B2, B3, B4, B5, B7, B8, B12, B13, B18, B19, B20, B26, B28 • LTE TDD B38, B39, B40, B41 2G/M1/NB1 world (myDatalogEASY IoT GW ATEX 2G/M1/NB1 World) • 2G GPRS 900MHz / 1800MHz • 2G GPRS 850MHz / 1900MHz • LTE B1, B2, B3, B4, B5, B8, B12, B13, B20, B25, B26, B28, B66, B85

SIM [2)]	The following options can be selected using the DeviceConfig configuration program: • Integrated SIM chip • SIM slot Additional information is provided in "Using the external SIM slot" on page 48

1) The number and assignment of the antenna connectors depends on the selected variant of the myDatalogEASY IoT GW ATEX .

2) In order for the SIM slot to be used, the chargeable feature "Activation code VPN SIM (300539)" must be unlocked or order option "Feature activation VPN SIM (300729)" is required.

Chapter 5 General specifications

The information in this manual has been compiled with great care and to the best of our knowledge. The manufacturer, however, assumes no liability for any incorrect specifications that may be provided in this manual. The manufacturer is not responsible for direct, indirect, accidental or consequential damages which arise from errors or omissions in this manual even if advised of the possibility of such damages. In the interest of continuous product development, the manufacturer reserves the right to make improvements to this manual and the products described in it at any time and without prior notification or obligation.

> ***Note:*** *The specifications in this manual are valid as of the versions listed on the front page. Revised versions of this manual, as well as software and driver updates are available in the service area of the myDatanet server.*

5.1 Translation

For deliveries to countries in the European Economic Area, the manual must be translated into the language of the respective country. If there are any discrepancies in the translated text, the original manual (German) must be referenced or the manufacturer contacted for clarification.

5.2 Copyright

The copying and distribution of this document as well as the utilisation and communication of its contents to others without express authorisation is prohibited. Contraventions are liable to compensation. All rights reserved.

5.3 General descriptive names

The use of general descriptive names, trade names, trademarks and the like in this manual does not entitle the reader to assume they may be used freely by everyone. They are often protected registered trademarks even if not marked as such.

5.4 Ex protection

The measurement device myDatalogEASY IoT GW ATEX is designed for use in areas with a zone 2 explosive atmosphere.

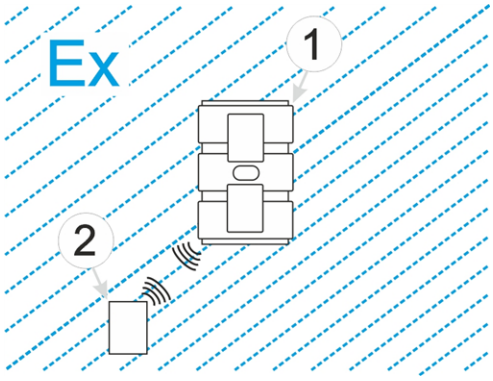
The following conditions must be observed:

- During installation of the device and any components connected to the device, it must be ensured that the requirements for planning, selection, construction and initial inspection of electrical installations in areas with explosive atmosphere as per IEC 60079-14 are met.
- Only the manufacturer's original spare parts must be used.
- During installation, it must be ensured that the device is protected against light (e.g. day light, artificial lighting) and it must be installed in such a way that the light resistance requirements as per IEC60079-0 are met.
- The installation must be executed in such a way that there is no risk of ignition caused by electrostatic charges during intended use, maintenance and cleaning.
- The device may only be mounted or installed if no explosive atmosphere is present. It must be ensured that the protection level (voltage), clearances, isolation distances, creepage distances and comparative tracking index are adhered to.
- Do not separate the connectors, when energized or when there may be an explosive atmosphere.
- When using an external supply or charging voltage, measures must be implemented outside the device to ensure transient protection (as per EN 60079-7) that will ensure the maximum permissible operating voltage is not exceeded by more than 40%. This requirement is met with the "2-pin connection cable for supply, 5 m" (301192) (see "Cable" on page 282).
- The housing may only be opened if no explosive atmosphere is present.
- The device may only be operated with power supply units PSU713 BP or PSU DC.
- The power supply unit may only be replaced if no explosive atmosphere is present.
- Only the antennas, tested and approved by the manufacturer, may be connected to the antenna connectors (see "Antennas" on page 281).
- Possible danger of electrostatic discharge build-up - clean the device with a moist or electrostatically dissipative cloth.
- A mounting distance of min. 50 mm must be maintained on all sides to prevent any soiling.
- The device may only be installed in an environment that meets contamination level 2 or lower in accordance with EN 60664-1.

II 3G Ex ec nC IIC T4 Gc

Important note: *The Ex approval is only valid if the corresponding marking is provided on the type plate of the measurement instrument .*

Important note: *The certificates of conformity and any relevant test certificates from the respective authorities must be carefully observed during installation and commissioning.*



1 myDatalogEASY IoT GW ATEX	**2** BLE sensor approved for Ex zone

5.5 Safety instructions

For the connection, commissioning and operation of the myDatalogEASY IoT GW ATEX , the following information and higher legal regulations of the country (e.g. ÖVE), such as valid EX regulations as well as the applicable safety and accident prevention regulations for the respective application case must be observed.

Read this manual completely before unpacking, setting up or operating this device. Observe all hazard, danger and warning information. Non-observance can lead to serious injuries to the operator and/or damage to the device.

Ensure that the safety equipment of this measurement instrument is not impaired. Install and use the measurement system only in the manner and method described in this manual.

> ***Important note:*** *The manufacturer's products that are designed for outdoor use include extensive protection against moisture and dust penetration. Connectors that are not used must be covered with protective caps to protect against penetrating moisture and dust. The operator is responsible for complying with local safety regulations.*

5.5.1 Use of the hazard warnings

DANGER:
Indicates a potential or threatening hazardous situation that will result in death or serious injuries if not avoided.

WARNING:
Indicates a potential or threatening hazardous situation that can result in death or serious injuries if not avoided.

CAUTION:
Indicates a potential hazardous situation that can result in minor or moderate injuries or damage to this instrument.

Important note: *Indicates a situation that can result in damages to this instrument if it is not avoided. Information that must be particularly emphasised.*

Note: *Indicates a situation that does not result in any injury to persons.*

Note: *Information that supplements the specifications in the main text.*

5.5.2 General safety instructions



WARNING:
Hazardous electric voltage can cause electric shock or burns. Always switch off all of the used power supplies for the device before installing it, completing any maintenance work or resolving any faults.



WARNING:
Ensure that the device is fully deactivated and cannot activate automatically when sending/returning it as air freight. Information on this is provided in chapter "Storage of the product" on page 33. If you have any unanswered questions, contact the manufacturer (see "Contact information" on page 293).



WARNING:
Never use this device in areas where the use of wireless equipment is prohibited. The device must not be used in hospitals and/or in the vicinity of medical equipment, such as heart pacemakers or hearing aids, as their functionality could be compromised by the GSM/GPRS modem contained in the device.

5.5.3 Safety and preventative measures for handling GSM/GPRS modems

The following safety and preventative measures must be observed during all phases of installation, operation, maintenance or repair of a GSM/GPRS modem. The manufacturer is not liable if the customer disregards these preventative measures.



CAUTION:
The mobile radio connection must not be used in hazardous environments.

No guarantee of any kind, whether implicit or explicit, is given by the manufacturer and its suppliers for the use with high risk activities.

In addition to the following safety considerations, all directives of the country in which the device is installed must be complied with.

> ***Important note:*** *No liability shall be assumed at any time and under no circumstances for connections via a GSM/GPRS modem for which wireless signals and networks are utilized. The GSM/GPRS modem must be switched on and operated in an area where sufficient signal strength is present.*

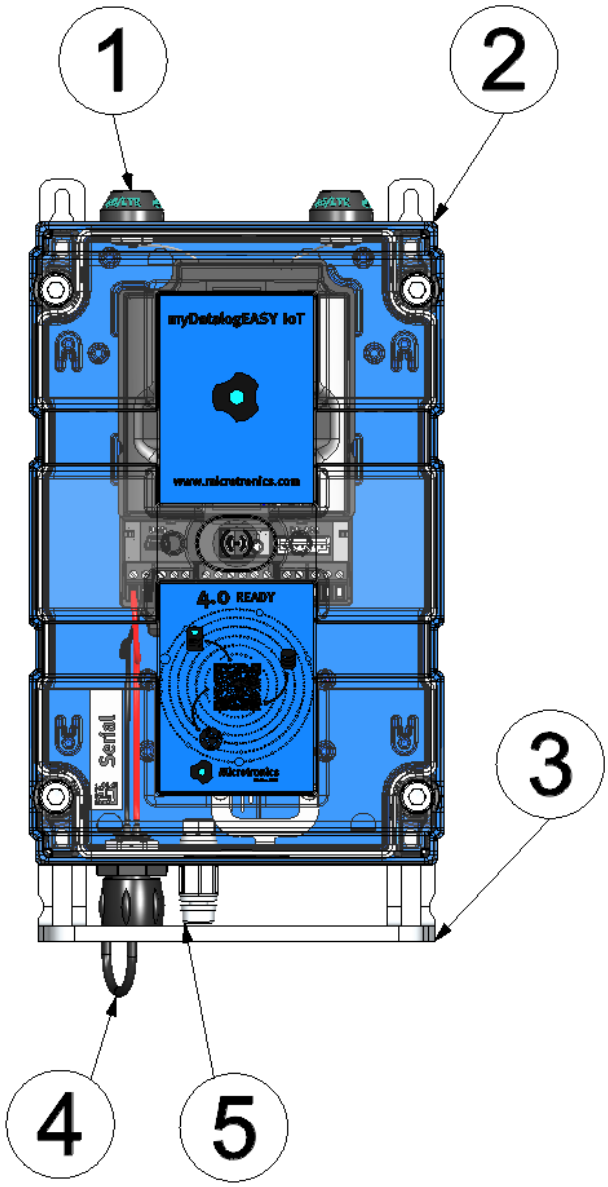
5.5.3.1 Safety and precautionary measures for the GSM/GPRS modem installation

- This device must only be installed by a trained technician who applies the recognised installation practices for a radio frequency transmitter including the correct grounding of external antennas.
- The device must not be operated in hospitals and/or in the vicinity of medical equipment such as heart pacemakers or hearing aids.
- The device must not be subjected to strong vibrations or impacts.
- The GSM/GPRS modem can cause interferences if it is located in the vicinity of television sets, radios or computers.
- Do not open the GSM/GPRS modem. Any modification to the device is prohibited and will result in the operating licence being revoked.
- The use of GSM services (SMS messages/data communication/GPRS, etc.) may incur additional costs. The user is solely responsible for any resulting damages and costs.
- Do not install the device in any other way to the one described in the operating instructions. Improper use will invalidate the warranty.

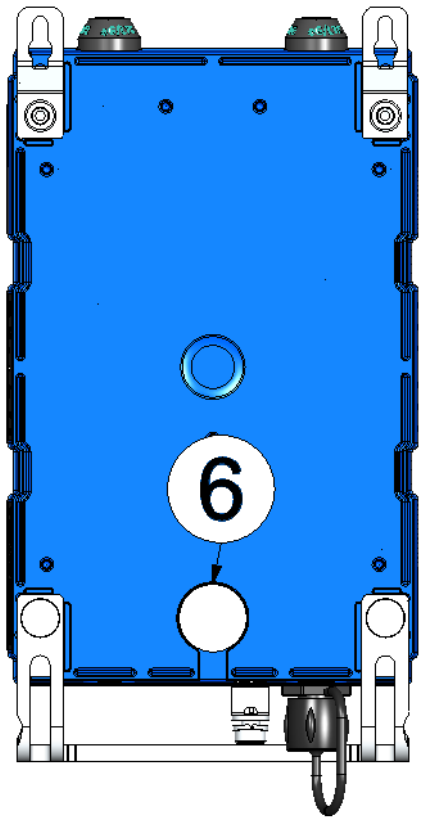
5.5.3.2 Safety measures for installing the antenna

- Only use antennas that are recommended or supplied by the manufacturer.
- The antenna must be installed at a distance of at least 20 cm from individuals.
- The antenna must not be extended outside protected buildings and must be protected against lightning strikes.
- The voltage supply must be switched off before replacing the antenna.

5.6 Overview



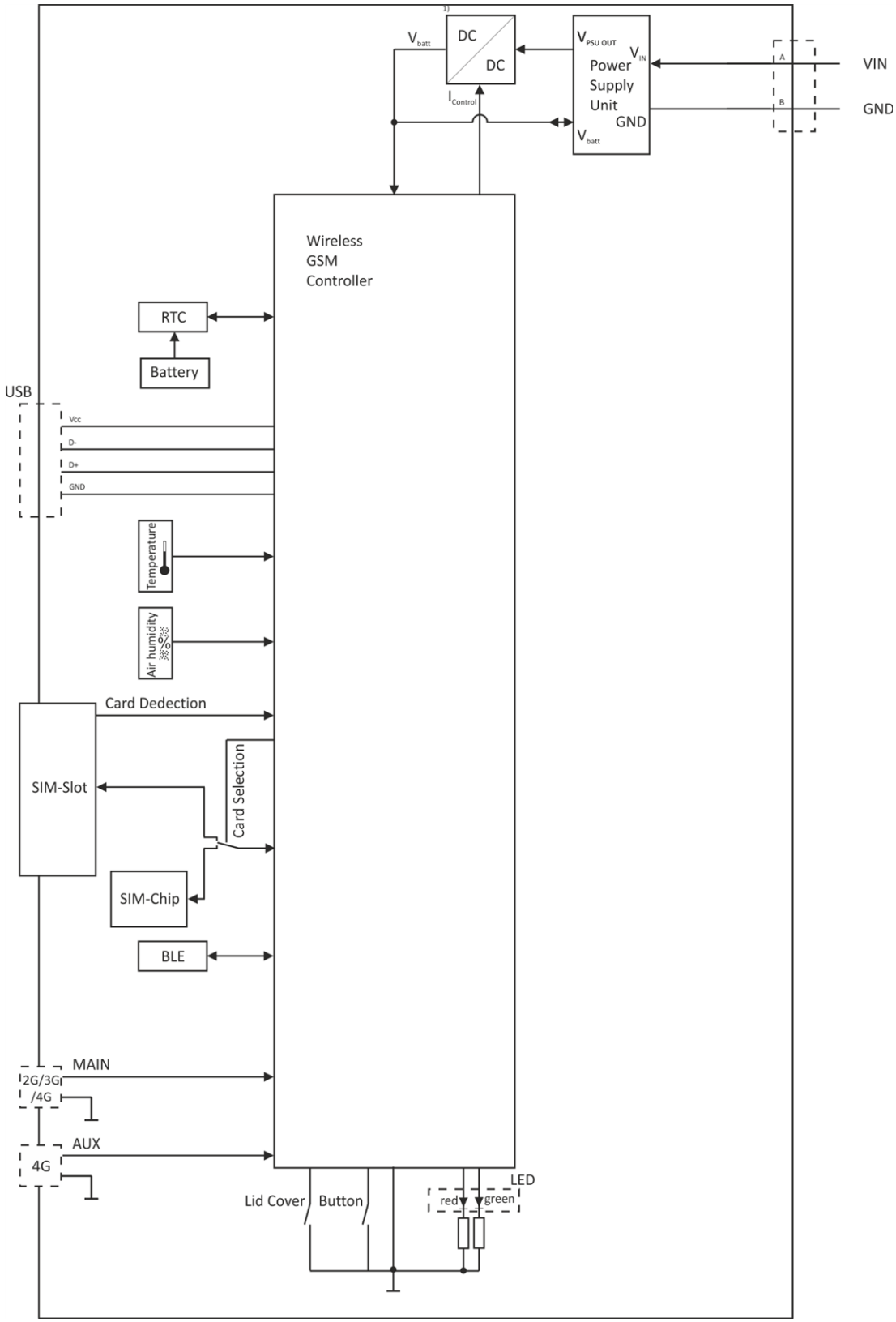
Front of the myDatalogEASY IoT GW ATEX



Rear of the myDatalogEASY IoT GW ATEX

1	Antennenanschlüsse (bis zu 2 x FME-M)	**4**	Connector for supply or charging voltage
2	Housing cover	**5**	Earth connection (Minimum requirement for the earth cable: 4 mm²)
3	Metal bracket to protect the connectors	**6**	Pressure compensation

5.6.1 Block diagram



Block diagram of the myDatalogEASY IoT GW ATEX

[1] It is a DC/DC converter with controllable output current.

Note: *Detailed block diagrams of the most common power supply units are provided in chapter "Technical details about the energy supply" on page 76.*

5.7 Intended use

The portable, freely programmable gateway is designed for determining, processing and transferring measurement data acquired via a Bluetooth connection (Bluetooth Low Energy). The device can operate without mains power. The measured and recorded data is stored on a non-volatile storage medium. This saved data can be sent to a central server via the mobile network for further processing or transferred to a PC via a Bluetooth connection (Bluetooth Low Energy). Using the software provided by the manufacturer, the data can then be transferred from the PC to a central server. The device is equipped with an integrated SIM chip for establishing a mobile connection. The maximum permissible limit values specified in chapter "Specifications" on page 19 must be observed. The manufacturer shall not be liable for any operational cases that deviate from these limit values and have not been approved by the manufacturer in writing.

Note: *This device is exclusively intended to be used for the purposes as described before. Any other use or use beyond what is specified or a modification of the device shall be deemed to be not for the intended purpose and is not permitted without the express written consent of the manufacturer. The manufacturer shall not be held liable for any damages that may result from such unauthorised use or modification. The operator alone bears the associated risk.*

Note: *The manufacturer is not liable for data loss of any kind.*

Note: *The integrated SIM chip provides a mobile communications connection to a variety of international service providers. In order to be able to utilise all functions of the device, you must ensure that the device is located in the service area of one of these service providers. You can find a list of all supported countries and associated service providers under www.microtronics.com/footprint. A Managed Service contract with Microtronics Engineering GmbH is required for use of the mobile data transmission (see www.microtronics.com/managedservice). This includes the provisioning of the mobile communications connection via the network of the service provider included in the above-mentioned list.*

5.8 General product information

The device is a compact, freely programmable gateway for determining, processing and transferring measurement data. For measurement data acquisition, a Bluetooth interface (Bluetooth Low Energy) is available. The myDatalogEASY IoT GW ATEX is also equipped with internal sensors (internal housing temperature and air humidity in the housing).

The customer can create their own application via the rapidM2M Studio (see "rapidM2M Studio " on page 125). During development the part of the application that needs to be installed on the device (i.e. the device logic) is loaded into the myDatalogEASY IoT GW ATEX via the USB interface. For applications that are provided via the rapidM2M Store installation of the device logic is performed via the mobile network or Bluetooth connection in the course of connecting the site with the myDatalogEASY IoT GW ATEX . The device logic enables access to the Bluetooth interface (Bluetooth Low Energy) thus providing the customer with the option to connect to almost all of the devices and sensors that are compatible with these interfaces and to implement the corresponding communication protocols.

The myDatalogEASY IoT GW ATEX provides the customer with a memory area for their data (3MB) as well as 10 independent memory blocks each with 4000 Bytes for the configuration data. In addition to the 4 registration memory blocks each with 1kB , that are saved in the flash, the myDatalogEASY IoT GW ATEX has another one that can optionally be initialised via the "rM2M_RegInit()" function and that is saved in the

RAM. Its size can be specified during initialisation, although it is limited to a maximum of 1kB . The registration memory blocks are assigned to predefined purposes and are designed for storing device-specific data (see "Registration memory blocks" on page 45).

Recorded measurement data can be sent to a central myDatanet server via the mobile network for further processing or read out locally via a Bluetooth connection (Bluetooth Low Energy). The DeviceConfig configuration program provided by the manufacturer (see "DeviceConfig " on page 91) and the USB BLE-Adapter (300685) or a smartphone compatible with Bluetooth Low Energy and the "tbd" smartphone app are required for reading via a Bluetooth connection. Both the DeviceConfig configuration program and the "tbd" smartphone app offer the possibility to forward the measurement data to a central myDatanet server. The device is equipped with an integrated SIM chip for establishing a mobile connection.

Synchronisation of the configuration and registration data with the server is also possible via the mobile network or the Bluetooth connection (Bluetooth Low Energy). The customer holds the responsibility for initialising the connection to the central myDatanet server via the mobile network connection (see "rM2M_ TxStart()"). However, the system automatically synchronises the configuration, registration and measurement data with the server.

5.9 Device labelling

The specifications in this user manual apply exclusively to the myDatalogEASY IoT GW ATEX device type. The type plates are located on the right side of the device and contains the following specifications:

- Pin assignment of the connector for supply or charging voltage
- Voltage supply specifications
- List of suitable power supply units
- Hazard note regarding electrostatic charge build-up
- Type designation
- Serial number
- Item number
- Country list profile of the SIM chip
- Hardware revision
- Environmental conditions during operation
- Protection class
- Week and year of production
- Ex protection designation as specified in chapter "Declaration of conformity" on page 13
- CE marking
- Logo for the EU WEEE Directive
- Chargeable features that were unlocked at the time of delivery
- Name and address of the manufacturer

The correct specification of the type designation and serial number is important for all queries and spare part orders. Only then can we process requests promptly and properly.



Type plates myDatalogEASY IoT GW ATEX

Note: *This symbol indicates the country list profile (see www.microtronics.com/footprint) of the SIM chip installed in the device.*

Note: *These operating instructions are part of the device and must be available to the user at all times. The safety instructions contained therein must be observed.*

WARNING:
It is strictly prohibited to disable the safety equipment or modify its mode of operation.

5.10 Installation of spare and wear parts

Be advised that spare and accessory parts that have not been supplied by the manufacturer have also not been inspected or approved by the manufacturer. The installation and/or use of such products can possibly have a negative impact on the specified constructional properties of the device. The manufacturer shall not be liable for any damages that arise from the use of non-original parts and non-original accessory parts.

> ***Note:*** *The use of spare and wear parts that are not approved by the manufacturer shall void the Ex approval.*

5.11 Storage of the product

For safekeeping of the myDatalogEASY IoT GW ATEX , ensure that all relevant data was transferred to the myDatanet server. If necessary, initiate a transmission directly on the device using the solenoid contact (see "Solenoid switch" on page 83), if you have included this in your Device Logic, and then check again to see whether all of the relevant data has now been transferred. If your device does not include the option to initiate the transmission of temporarily stored measurement data, you may have to wait until the next scheduled data transmission for all of the data to be sent to the myDatanet server. This particularly applies to the "interval" connection type (see "rM2M_TxSetMode()"). If the "Interval & wakeup" connection type has been selected, you can initiate the transmission via the myDatanet server. Then check again to make sure all of the relevant data has been transferred. With the "Online" connection type, the determined measurement data is immediately transferred to the myDatanet server. The data on the server is always up-to-date and the device can be switched off at any time. Remove the power supply unit before disconnecting the cables and antenna. If possible, switch off the supply or charging voltage before disconnecting the cable from the connector for the supply or charging voltage (see "Connection of the power supply" on page 71). Store the myDatalogEASY IoT GW ATEX and power supply unit in the original packaging.

The configuration and most recently determined data are retained. The system time also continues to run thanks to the hardware real-time clock equipped with its own buffer battery. This means that a valid time basis is available immediately following recommissioning (see "Technical details about the system time" on page 77).

5.12 Warranty

The device has been functionally tested before delivery. If it is used as intended (see "Intended use" on page 30) and the operating instructions, the applicable documents(see "Applicable documents " on page 79) and the safety notes and instructions contained therein, are observed, no functional restrictions are to be expected and perfect operation should be possible.

> ***Note:*** *Please also note in this regard the next chapter "Disclaimer" on page 34.*

> ***Note: Limitation of warranty***
>
> *In the event of non-compliance with the safety instructions and instructions in this document, the manufacturer reserves the right to limit the warranty.*

5.13 Disclaimer

The manufacturer assumes no liability

- for damages owing to **a change** of this document. The manufacturer reserves the right to change the contents of this document and this disclaimer at any time and without any notice.
- for damages to persons or objects resulting from **failure to comply** with applicable **regulations**. For connection, commissioning and operation of the devices/sensors all available information and higher local legal regulations (e.g. in Austria ÖVE guidelines) such as applicable Ex regulations as well as safety requirements and regulations in order to avoid accidents shall be adhered to.
- for damages to persons or objects resulting from **improper use**. For safety and warranty reasons, all internal work on the instruments beyond from that involved in normal installation and connection, must be carried out only by qualified Microtronics personnel or persons or companies authorised by Microtronics .
- for damages to persons or objects resulting from the use of instruments in technically **imperfect** condition.
- for damages to persons or objects resulting from the use of instruments **not in accordance with the requirements**.
- for damages to persons or objects resulting from **failure to comply** with **safety information** contained within this instruction manual.
- for missing or incorrect measurement values or resulting consequential damages due to **improper installation**.

5.14 Obligation of the operator



WARNING:
In the EEA (European Economic Area), the national implementation of the framework directive (89/391/EEC) as well as the associated specific directives and from these in particular, the directive (2009/104/EC) about the minimum safety and health requirements for use of work equipment by workers at work, each in their respective version are to be complied with.

The operator must obtain the local operating licence and the associated documents.

In addition, the operator must comply with the local legal requirements for

- the safety of the personnel (accident prevention measures),
- the safety of the equipment (protective equipment and maintenance),
- the product disposal (waste disposal law),
- the material disposal (waste disposal law),
- the cleaning (cleaning agents and disposal) and
- the environmental protection amendments.

Before commissioning, the operator must ensure that the installation and commissioning – provided these were performed by the operator himself – are in compliance with the local regulations.

5.15 Personnel requirements

Installation, commissioning and maintenance may only be completed by personnel who meet the following conditions:

- Qualified specialist personnel with the relevant training
- Authorised by the facility operator

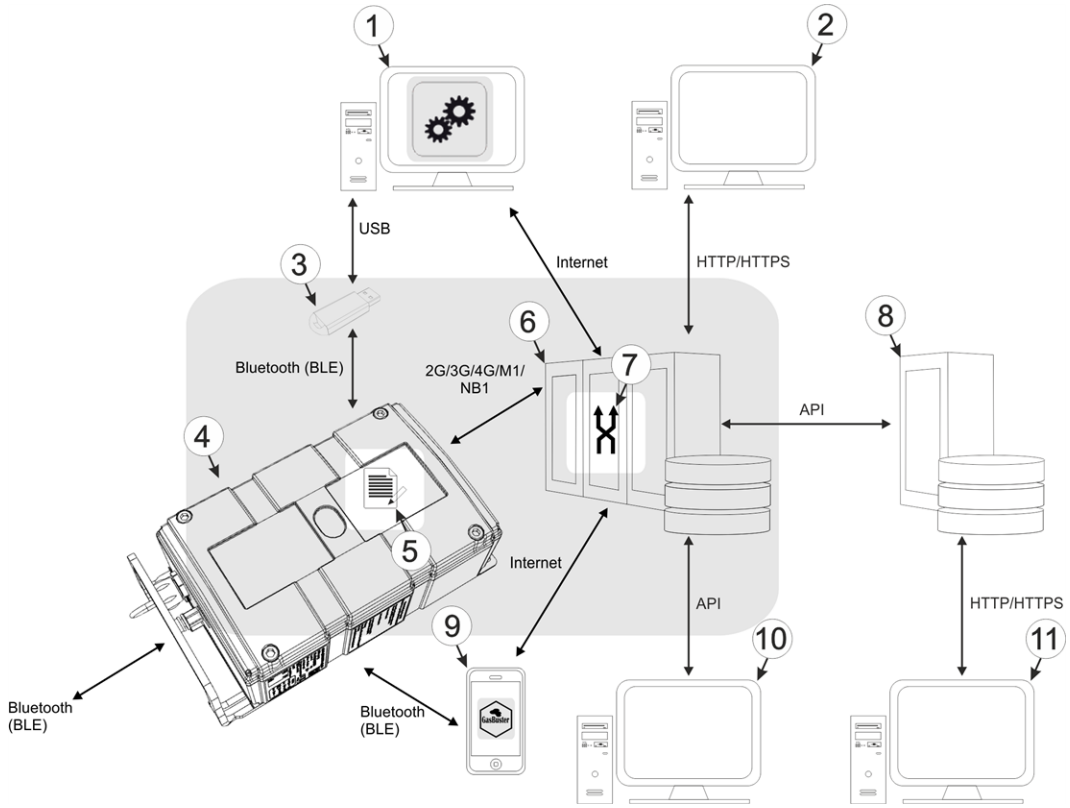
Note: Qualified personnel

In the context of these instructions and the warnings on the product itself, individuals responsible for the setup, installation, commissioning and operation of the product must have gained relevant qualifications relating to their activities, including, for example:

- *Training, instruction and authorisation to activate/deactivate, ground and label electric circuits and devices/systems in accordance with the standards of safety engineering.*
- *Training or instruction on the maintenance and use of suitable safety equipment in accordance with the standards of safety engineering.*
- *First aid training*

Chapter 6 Functional principle

In the graphic below, all of the components that are part of the myDatanet are illustrated in grey. All other components must be provided/created by the customer.



Functional principle

1	PC with the DeviceConfig configuration program installed
2	Client that accesses the interface of the myDatanet server via the web browser
3	USB BLE-Adapter (Bluetooth Low Energy to USB converter)
4	myDatalogEASY IoT GW ATEX with integrated managed service SIM chip (including data transmission)
5	Application created by the customer (device logic) that collects and records the data (see "Device Logic" on page 131)
6	myDatanet server to which the data is transferred
7	Data Descriptor defined by the customer that enables the use of the measurement data and configurations generated by the application (device logic) in connection with the interface of the myDatanet server (see "Data Descriptor " on page 251)
8	Customer-specific server that provides clients with their own interface. The customer-specific server obtains the data via the API interface of the myDatanet server (see "API" on page 263).
9	Smartphone with "tbd" smartphone app installed
10	Client, on which a PC program is running, that obtains its data via the API interface of the myDatanet server (see "API" on page 263)
11	Client that uses a web browser to access the interface of the customer-specific server, which receives its data by means of the API interface of the myDatanet server.

As illustrated in the previous figure (see "Functional principle" on page 37), there are 3 options for transferring the data from the myDatalogEASY IoT GW ATEX to the myDatanet server:

- Directly via 2G/3G/4G/M1/NB1 mobile connection
- Indirectly, by initially reading the data from the device via the DeviceConfig configuration program using the Bluetooth connection (BLE) and then using the PC's Internet connection to transfer the data to the server
- Indirectly, by reading the data from the device via the "tbd" smartphone app using the Bluetooth connection (BLE) and using the smartphone's Internet connection to transfer the data to the myDatanet server

Functions and components provided by myDatanet :

- myDatalogEASY IoT GW ATEX

  Programmable (see "Device Logic" on page 131), portable device with integrated memory and Bluetooth interface (Bluetooth Low Energy) to connect sensors and devices to the myDatanet server (2G/3G/4G/M1/NB1).

- USB BLE-Adapter (300685, optional accessories)

  This hardware module is connected directly to the USB interface of the PC. The required drivers are included in the installation package of the DeviceConfig configuration program. Detailed information on this is provided in chapter "Installing USB BLE-Adapter driver" on page 95.

- DeviceConfig configuration program (optional)

  The DeviceConfig configuration program is required to read out the data from the myDatalogEASY IoT GW ATEX via the Bluetooth connection (Bluetooth Low Energy) and to forward it to a central myDatanet server. An Internet connection is required for this purpose. The DeviceConfig configuration program uses port 51241 to transfer data.

- "tbd" smartphone app (optional)

  When combined with a Bluetooth Low Energy compatible smartphone, the "tbd" smartphone app provides the option of reading the data from the myDatalogEASY IoT GW ATEX via the Bluetooth connection (Bluetooth Low Energy) and transferring it to a central myDatanet server.

- Managed Service

  Managed Service is the basis for operating the devices and provides a wide range of services. Managed Service includes updates for device firmware, mobile data transmission on a global scale and free support - providing you with one contact person for the entire solution.

- myDatanet server

  Database for saving the measurement data and configurations. Data is either accessed via the API of the server (see "API" on page 263) or web interface of the server.

Functions and components provided by the customer

- Sensors and devices

  Sensors and devices that have a Bluetooth interfaces that are compatible with the specifications listed in the chapter "Specifications" (see "Specifications" on page 19)

- Application (device logic)

  The firmware of the myDatalogEASY IoT GW ATEX only manages the synchronisation of the measurement data and configurations between the myDatalogEASY IoT GW ATEX and myDatanet server. The application created by the customer must record the measurement values and create the data blocks that are to be saved. The data blocks, on the other hand, are stored by the firmware of the myDatalogEASY IoT GW ATEX (see "rM2M_RecData()"). The time of the synchronisation and the type of connection must also be determined by the application created by the customer. Both of the API functions "rM2M_TxStart()" and "rM2M_TxSetMode()" are provided for this purpose.

- Data Descriptor

  The basic function of the myDatanet server is limited to the synchronisation of the measurement data channels ("histdata0" - "histdata9") and configuration blocks ("config0" - "config9") between the myDatalogEASY IoT GW ATEX and server. The Data Descriptor defined by the customer must separate the measurement data channels and configuration blocks to the individual data fields. An explanation of this is provided in chapter "Data structure" on page 251.

- Customer-specific server with web interface for the clients (optional).

  It can be used to create an individual web interface for the clients. In this case the data is read from the customer specific server via the API interface (see "API" on page 263) on the myDatanet server gelesen.

6.1 Recommended procedure

6.1.1 Development of M2M/IoT application

It is recommended to start with the definition of the Data Descriptor (see "Data Descriptor " on page 251) when developing a M2M/IoT application. It specifies the various data structures (measurement data, configurations, etc.) that are valid for the Device Logic as well as the myDatanet server. The definitions of the Data Descriptor also apply for accessing the data of the myDatanet server via the API.

The information type should be taken into consideration when assigning the data to the relevant containers ("histdata0" - "histdata9" or "config0" - "config9"). The "histdata0" - "histdata9" containers should be used for time series. If there is measurement data that will be generated frequently and data that will only be generated rarely, it is recommended to use two different containers (e.g. "histdata0" for the frequently generated and "histdata1" for the rarely generated). Similar also applies to the configuration data, for which the "config0" - "config9" containers are provided. When it comes to the configuration data, it is recommended to take a grouping based on logical context into consideration in addition to the frequency of change.

> ***Note:*** *A well thought out selection of the containers helps to reduce the required data volume and associated costs.*

6.2 Functionality of the internal data memory

Structure	Circular buffer
Total size	3MB
Number of sectors	8
Sector size	393.216 Byte

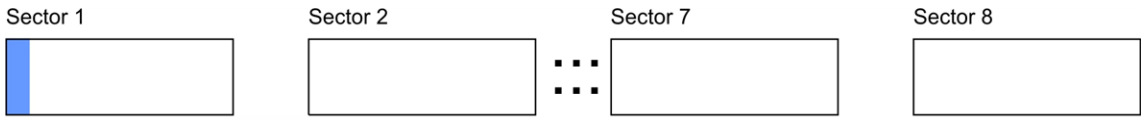
The internal data memory of the myDatalogEASY IoT GW ATEX is designed as a circular buffer with 8 sectors. If the entire memory (3MB) is full, the sector with the oldest data is deleted fully before new data can be saved in this sector again. This means that the internal data memory comprises at least 2,625 MB of valid data and a maximum of 3MB .

For this reason, it is recommended to coordinate the data transmission and record interval in such a way that a maximum of 2,625 MB has to be recorded between two transmissions. If it can be expected that individual transmissions fail due to poor network coverage, this must also be taken into consideration when calculating the data volume to be saved. Additionally, it must be noted that the system-related overhead is 11 Byte per data record and that the first 8 Byte of each sector are reserved for the internal memory management. The 11 Byte overhead already includes the timestamp, so it does not have to be taken into account when calculating the size of the entire record. If there is not enough free space in a sector to save the entire data record, the data record is written to the next sector. This means that a data record is not written over the sector limits.

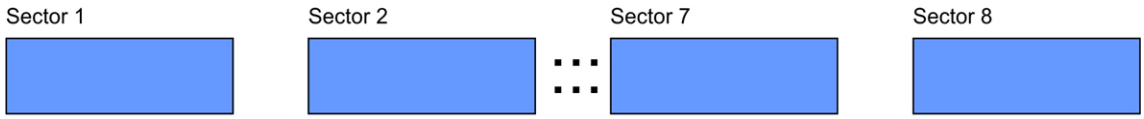
Note:

Additional explanation regarding the functionality of the circular buffer

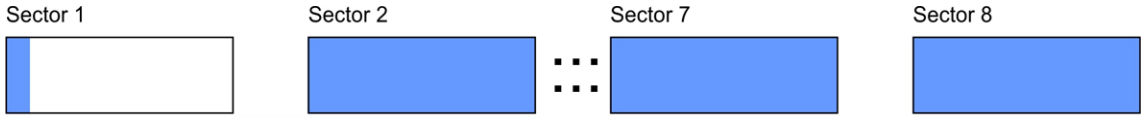
Data memory after the first data is recorded:



Data memory once 3MB has been recorded



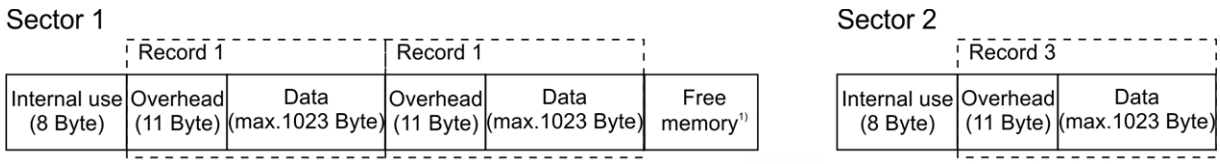
Data memory if further data is recorded once 3MB has already been recorded



Note:

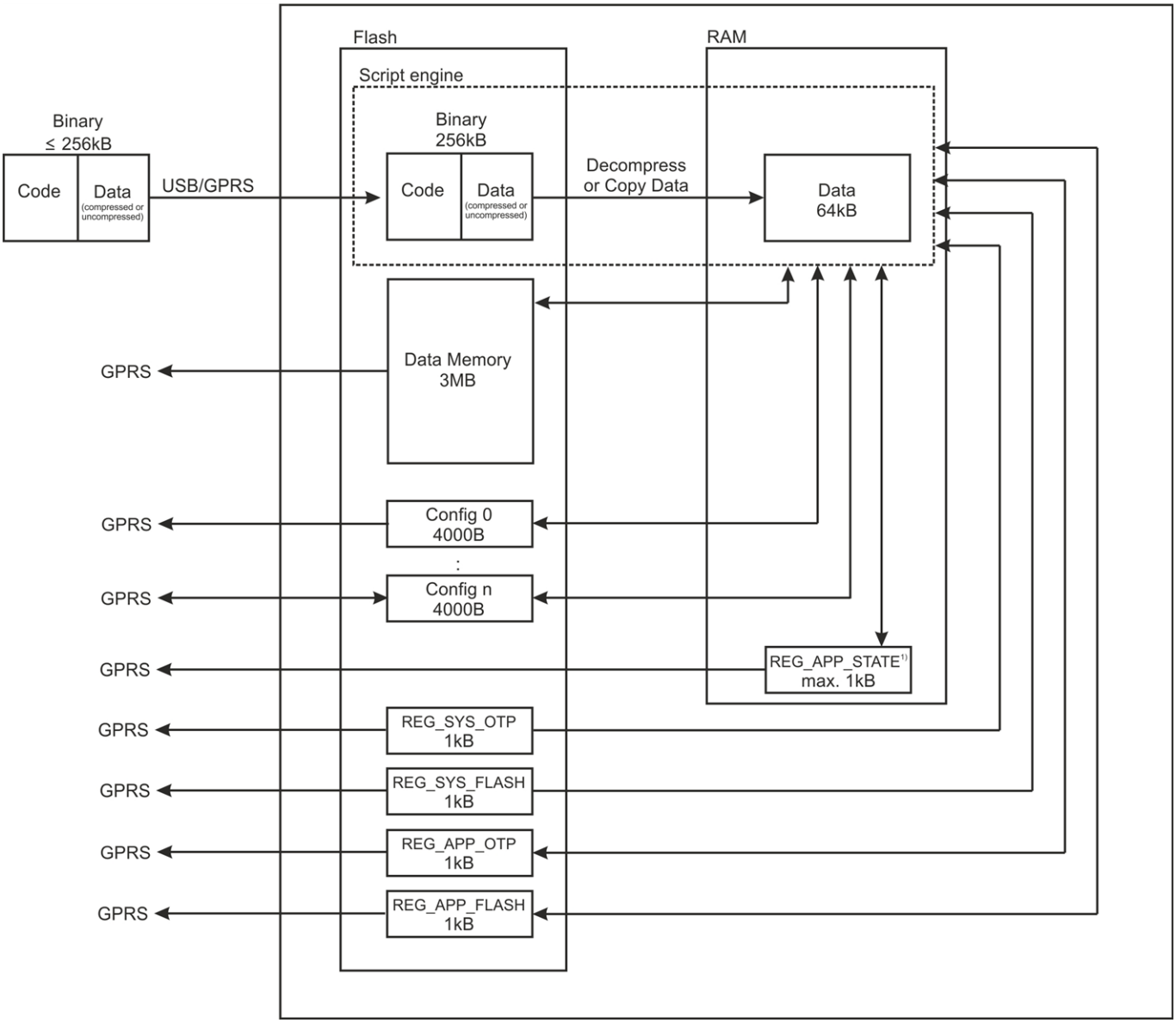
Additional explanation on calculating the data volume to be saved:

To provide a clear and simple overview, the following example assumes that the sectors can only record two complete data records.



1) Free memory in sector 1 is not enough to record a full data record (overhead + data).

6.3 Memory organisation



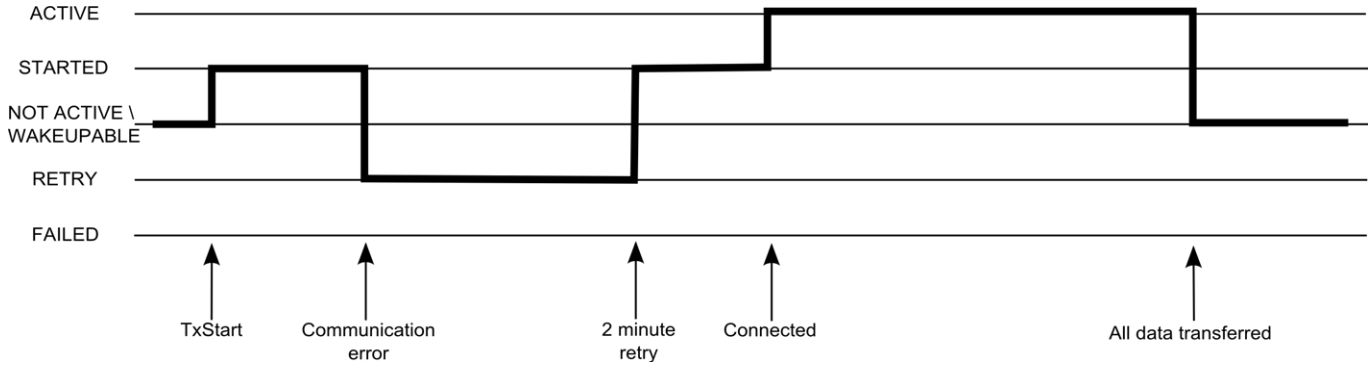
Organisation of the myDatalogEASY IoT GW ATEX memory

1)This memory block is only available if it was initialised via the "rM2M_RegInit()" function.

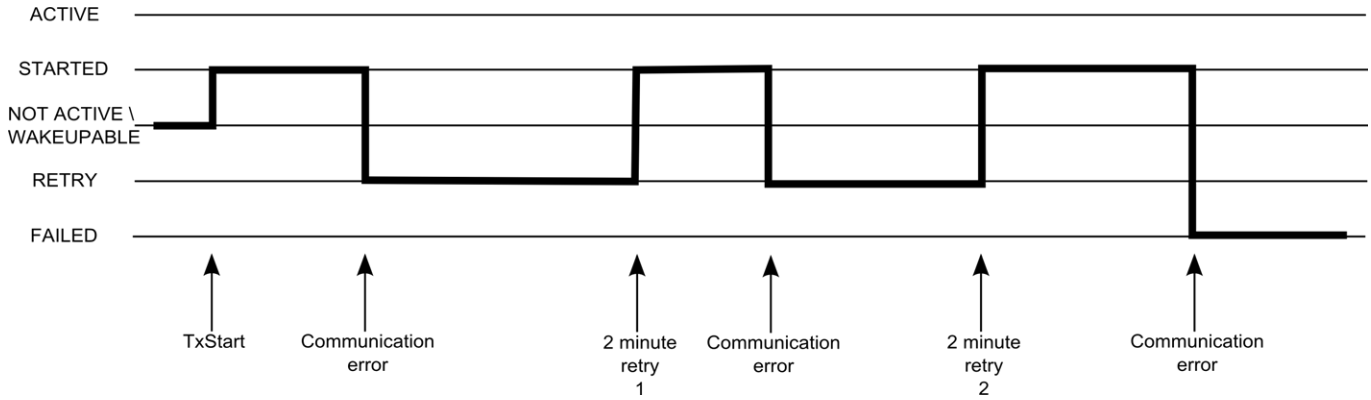
The size of the binary must not exceed 256kB for the transmission to the myDatalogEASY IoT GW ATEX . If necessary, the data area of the binary can be compressed using a compiler instruction (#pragma amxcompress <0-3>). The binary (256kB), 10 configuration blocks (4000 Bytes each), 4 registration memory blocks (1kB each) and measurement data (3MB) are stored in the flash memory of the myDatalogEASY IoT GW ATEX . To execute via the script engine, the data area of the binary is decompressed, if necessary, and copied to the RAM. The maximum size for the decompressed data area of the binary in the RAM is 64kB . The optional registration memory blocks (e.g. REG_APP_STATE) which can be initialized via the "RM2M_RegInit ()" function are also stored in RAM.

6.4 Procedure in case of connection aborts

If the connection is terminated, another attempt to establish a connection is made after 2min. for all connections, except for "online" mode. Up to 2 attempts to establish a connection are made.



Connection could be established during first retry.



Connection could not be established despite 2 retries.

ACTIVE

Connection to the myDatanet server established. Data is being transmitted.

STARTED

Connection establishment initiated.

NOT ACTIVE

The system is waiting for the next connection establishment to be initiated. The last connection attempt was successful and all of the data was transmitted.

WAKEUPABLE

The modem is logged into the GSM network and the system waits for the next connection attempt to be initiated. The last connection attempt was successful and all of the data was transmitted. As the modem is logged into the GSM network, the connection establishment can also be initiated via the myDatanet server (see "myDatanet Server Manual " 805002).

RETRY

The system waits 2min. until the next attempt to establish a connection.

FAILED

The system is waiting for the next connection establishment to be initiated. During the last connection attempt no data or not all of the data was transmitted.

> ***Note:*** *Depending on the type of communication error, the system may be restarted (e.g. to reinstall the SIM chip) before the "FAILED" status is set.*

The current connection status can be read out at any time via the "rM2M_TxGetStatus()" function.

6.4.1 Connection abort in "online" mode

A single direct connection attempt is made if the connection is lost in "online" mode. If it is not possible to establish a connection, 2 further attempts of the standard retry sequence are made at intervals of 2min. . If the connection was not established during the last retry, the device remains offline until the next connection attempt is triggered via the "rM2M_TxStart()" function.

6.4.2 Connection abort during a Device Logic download

The device reacts to a connection abort during the Device Logic download with the standard retry sequence (2 connection attempts at intervals of 2min.). In addition to this, the "SCRIPT_ERR, SCRIPT DOWNLOAD ERROR" log entry is entered in the device log as soon as the device has detected the connection abort. Although this does not affect the existing Device Logic. It can continue to be executed.

6.5 Timeout monitoring in online mode

In online mode, the myDatalogEASY IoT GW ATEX sends a keep alive ping to the myDatanet server by default at an interval of 15 min. and 3 sec. (i.e. every 903 sec.). This enables the server to detect whether the connection to the myDatalogEASY IoT GW ATEX is still available. To be able to detect interruptions to the connection more promptly, the standard interval for the keep alive ping can be adjusted via the "rM2M_ SetTCPKeepAlive()" function.

The "Bidirectional alive ping" can be activated on the server to ensure that the myDatalogEASY IoT GW ATEX can also promptly detect an interruption to the connection. This bidirectional alive ping can be activated globally for the complete server, for a specific customer or for a single site (see "myDatanet Server Manual " 805002). If the bidirectional alive ping is enabled, the myDatanet server sends a corresponding response to every keep alive ping from the device (keep alive response). Following receipt of the first keep alive response, the myDatalogEASY IoT GW ATEX will expect to receive a regular keep alive response within 10 sec. of the keep alive ping. If the keep alive response fails to appear three times in a row, an attempt is initially made to re-establish the communication without completely disconnecting the connection (i.e. by only reinitialising the connection on a TCP level only). Only once this has not worked will the myDatalogEASY IoT GW ATEX disconnect the connection to the myDatanet server completely and will immediately establish the connection again. The recording of the round trip time [ms] is also activated upon receipt of the first keep alive response. This means that the time until the keep alive response has been received by the server is measured for every subsequent keep alive ping. The "rM2M_TxItfGetStats()" function can be used to read the last determined "Round trip time" of the system.

6.6 Automatic selection of the GSM network

The GSM network to which the device should register must be selected, as the myDatalogEASY IoT GW ATEX is equipped with a SIM chip that provides a mobile connection via a variety of international service providers (see www.microtronics.com/footprint). This is completed automatically by the device.

6.7 Determining the GSM/UMTS/LTE signal strength

The internal update rate of the measurement value for the GSM/UMTS/LTE signal strength is dependent on the type of connection selected via the "rM2M_TxSetMode()" function:

- Interval: Updated during connection establishment
- Interval & Wakeup: Updated every 30 seconds
- Online: Updated every 5 seconds

The "rM2M_GSMGetRSSI()" function can be used to read the last determined value from the system.

6.8 Determining the GSM position data

An internal flag is set by the firmware every 24h , which ensures that the GSM position data is also determined the next time the "rM2M_TxStart()" function is called up. The positioning can however also be suppressed by setting the "RM2M_TX_SUPPRESS_ POSUPDATE" flag when calling up the function. It is also possible to trigger the determination of the GSM position data by setting the "RM2M_TX_POSUPDATE" flag when calling up "rM2M_TxStart()". If "Interval & Wakeup" connection mode was activated on the myDatalogEASY IoT GW ATEX , the previously described internal flag is set by the firmware and the connection establishment is triggered by the receipt of a Wakeup SMS (i.e. via the myDatanet server), the determination of the GSM position data is definitely executed and cannot be suppressed. The GSM position data cannot be generated if "online" connection mode is active.

6.9 Error handling

The following transmission mechanisms have been integrated in the firmware to ensure that problems with the Device Logic can be diagnosed and resolved remotely. In the event that there is no Device Logic, a connection to the myDatanet server is established every 24h . This backup interval is set to 1h if an existing Device Logic has been deactivated due to an error being detected by the system. The "Interval & wakeup" connection type is activated in both cases, which enables a connection to be initiated via the interface of the myDatanet (see "myDatanet Server Manual " 805002).

6.10 Registration memory blocks

In addition to 4 1kB blocks, that are saved in the flash, another one, that is saved in the RAM, can optionally be initialised via the "rM2M_RegInit()" function. Its size can be specified during initialisation, although it is limited to a maximum of 1kB . The registration memory blocks provide the option of storing device-specific data and synchronising it with the myDatanet server. The blocks only differs with regard to their access options and storage location. This results in predefined intended uses that are described in the following table:

Memory block	Access	Memory	Purpose
System-specific data			
REG_SYS_OTP [1)]	**readable:** Device Logic, myDatanet server	FLASH	System information that is written once as part of the production process
REG_SYS_FLASH [1)]	**readable:** Device Logic, myDatanet server	FLASH	System information that must be able to be changed during operation
Application-specific data			
REG_APP_OTP	**readable:** Device Logic, myDatanet server **writeable:** Device Logic	FLASH	Application-specific information that is written once as part of the production process (recommendation, writing it multiple times is not prevented by the firmware)
REG_APP_FLASH	**readable:** Device Logic, myDatanet server **writeable:** Device Logic	FLASH	Application-specific information that must be able to be changed during operation
Application-specific, volatile data			
REG_APP_STATE [2)]	**readable:** Device Logic, myDatanet server **writeable:** Device Logic	RAM	Application-specific information that must be able to be changed during operation and that does not require non-volatile storage in the flash (e.g. current device status).

1) Writing data in these two memory blocks is reserved for the manufacturer.

2) This memory block is only available if it was initialised via the "rM2M_RegInit()" function.

> ***Note:*** *It is also possible to write in the memory blocks as part of the production process via the local interfaces (USB and both UART). However, an agreement must be reached with the manufacturer to receive information about this (see "Contact information" on page 293).*

The "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()", "rM2M_RegDelValue()" and "rM2M_RegDelKey()" functions are available for accessing the registration memory blocks.

6.10.1 REG_APP_OTP

By saving the "Product Identity Profile" (PIP) in this registration memory block, the functions described in the following can be initiated on the myDatanet server. The PIP consists of the following fields:

pipCustomer

Name of the customer to whom the site should be assigned [2-50 characters].

pipCtx
Name of the site that should be created/used [2-50 characters].

pipAppId
ID of the IoT application based on which the site should be created [max. 50 characters].

pipAppVer (optional)
Version of the Device Logic currently installed on the device (e.g. 7) [Integer].

pipCtxAutocreate (optional)
Indicates whether the site (if it does not exist yet) should be created ("0" or "1" must be saved as the string)

- *"0": creation of a new site is not permissible*
- *"1": new site can be created (default)*

If the myDatanet server receives a PIP, the two basic scenarios are differentiated:

- There is no site with the name specified in the "pipCtx" field:

  The new site is only created if the customer and application template were found on the myDatanet server and "pipCtxAutocreate=1" or the field is missing.

- A site with the name specified in the "pipCtx" field was found on the server:

  In this case, the "pipCtxAutocreate" and "pipCustomer" fields are not relevant. The device is assigned to the site, even if it is located within a different customer, if the application ID in the "pipAppId" field and that of the found site match. The device is moved to the relevant customer for this purpose. The allocation of the existing device is deallocated if a device is already assigned to the site. The existing device is moved to the customer's pool.

6.11 File transfer

It is possible to register up to 60 files for the file transfer (see "FT_Register()"). A callback function, that should be called up when a file transfer command is received, must be transferred to the "FT_Register()" function during this process (see "Callback functions" on page 212). The callback function must be able to handle all file transfer commands (see "File transfer commands" in chapter "Constants" on page 212). The file properties must also be set via the "FT_SetProps()" function as part of the registration process. If a file should no longer be available for the file transfer, it can be removed from the registration using the "FT_Unregister()" function.

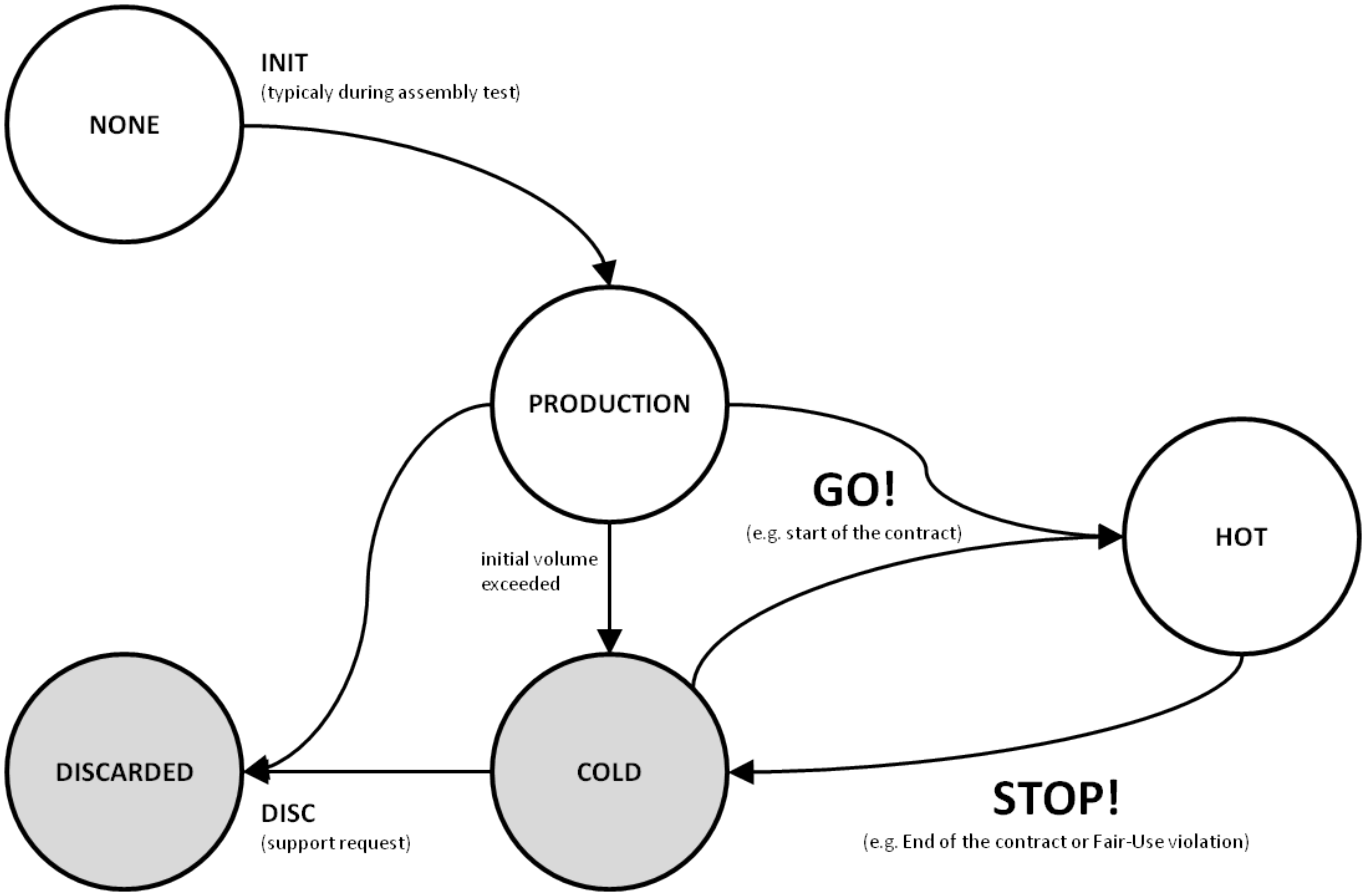
Upon receipt of a file transfer command, a session is started that is automatically terminated after 15sec. if the Device Logic is not handling the command correctly. Sessions can only be active one after another to prevent any conflicts.

6.12 Meaning of the SIM state

The device receives information about the permissible use of the SIM chip from the myDatanet server. The following states are defined for this SIM status:

SIM state	Transmission via Device Logic	Explanation
RM2M_SIM_STATE_NONE	Yes	Initial state
RM2M_SIM_STATE_PRODUCTION	Yes	New device is in stock
RM2M_SIM_STATE_HOT	Yes	Valid contract
RM2M_SIM_STATE_COLD	No	End of contract or violation of fair use policy
RM2M_SIM_STATE_DISCARDED	No	Device decommissioned

As the previous table illustrates, it is not possible to trigger the connection by Device Logic for the "RM2M_SIM_STATE_COLD" and "RM2M_SIM_STATE_DISCARDED" states. In this case, the functions "rM2M_TxStart()" and "rM2M_TxSetMode()" return error code "ERROR_SIM_STATE". Contact the manufacturer, to switch a device that is in the "RM2M_SIM_STATE_COLD" state back to the "RM2M_SIM_STATE_HOT" state (see "Contact information" on page 293). The SIM state can be read out at any time using the "rM2M_GSMGetInfo()" function. An entry is also added to the device log every time the SIM state changes (e.g. SIM_STATE, HOT).



State diagram of the SIM states

6.13 Using the external SIM slot

> ***Important note:*** *The chargeable "Activation code VPN SIM (300539)" feature must be released to be able to use the external SIM slot.*

The following two conditions must be met to activate communication via the external SIM card:

- The SIM card must be inserted in the external SIM slot (see "Inserting/replacing the SIM card" on page 56)
- The APN settings (APN, username and password) and the PIN code (if required by the SIM card) for the inserted SIM card must be transferred to the myDatalogEASY IoT GW ATEX using the DeviceConfig configuration program (see ""GSM" tab" on page 99) or must be set by the Device Logic via the "rM2M_SetExtSimCfg()" function.

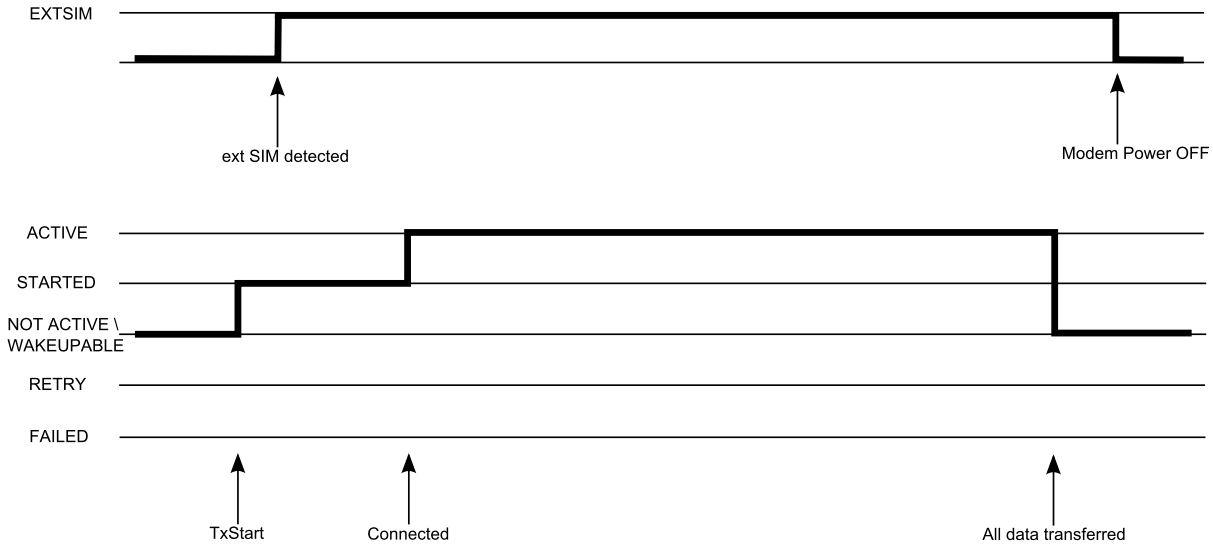
In the current implementation, using an external SIM card will not increase availability. This means that in the event of any communication problems relating to the external SIM card, the firmware will not automatically switch to the internal SIM chip. This cost-oriented approach helps to prevent any resulting charges for using the internal SIM chip as soon as the external SIM card has been activated.

To reactivate the internal SIM chip, it will not suffice to remove the external SIM card from the SIM slot. The APN settings must also be deleted from the myDatalogEASY IoT GW ATEX using the DeviceConfig configuration program or the Device Logic via the "rM2M_SetExtSimCfg()" function.

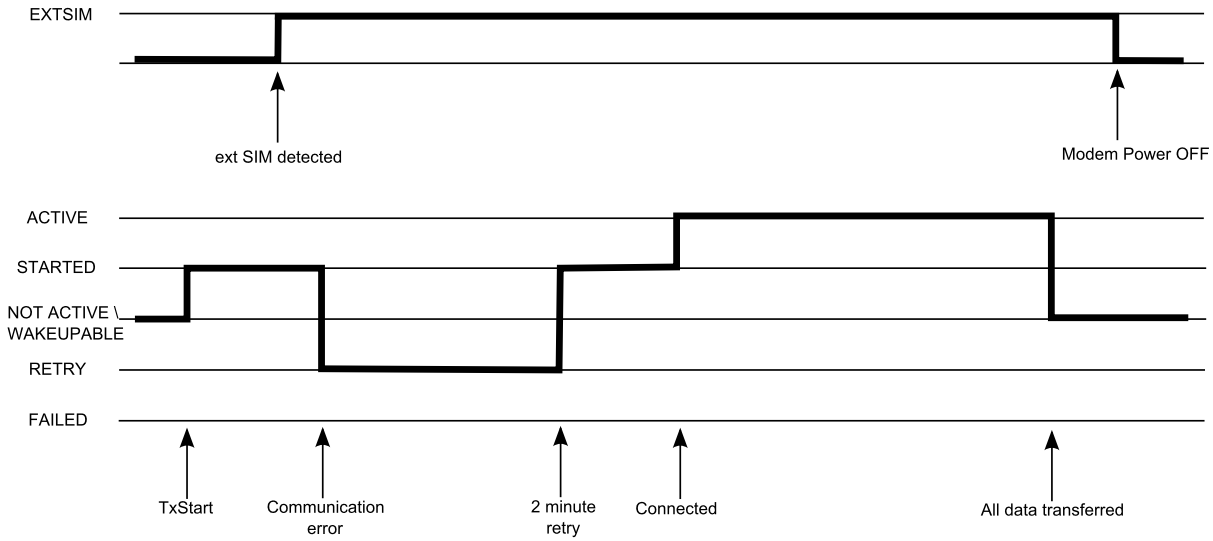
The following table specifies under which conditions the external SIM card or internal SIM chip is used. The parameters are checked each time the modem is activated."---" indicates a state where it is not possible to establish a connection.

External SIM slot released	External SIM card inserted	Correct APN setting saved in the device	SIM used
0	0	0	internal
0	0	1	---
0	1	0	---
0	1	1	---
1	0	0	internal
1	0	1	---
1	1	0	---
1	1	1	external

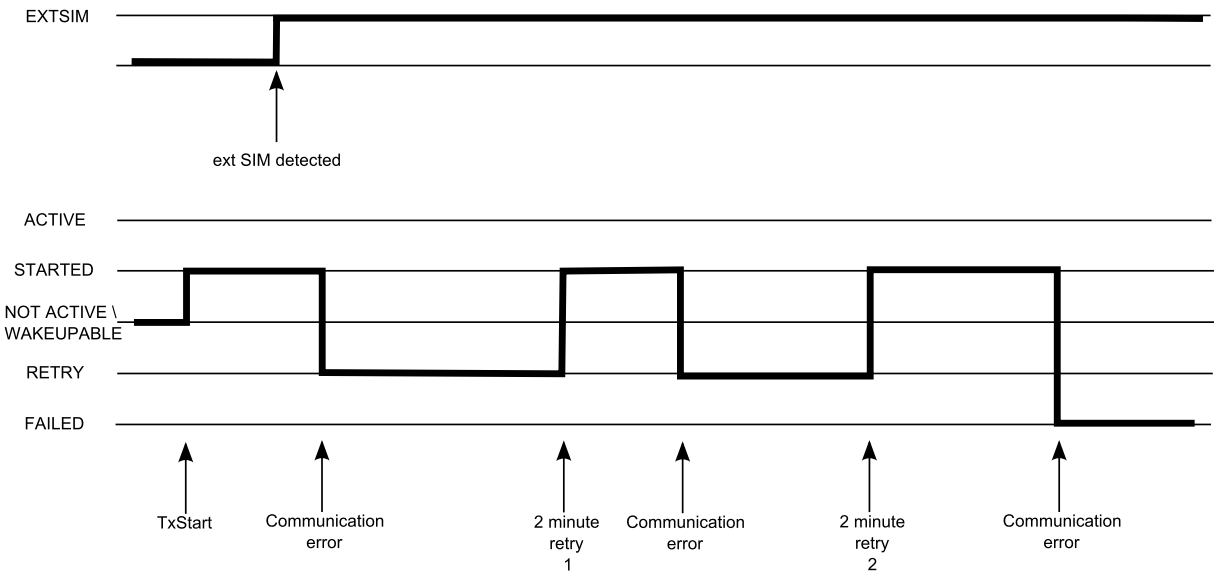
The signalling of the connection status described in chapter "Procedure in case of connection aborts" on page 42, is extended by the "EXTSIM" signal when using an external SIM card.



Connection could be established during first attempt.



Connection could be established during first retry.



Connection could not be established despite 2 retries.

EXTSIM

An external SIM card was detected following activation of the modem.

> ***Note:*** *To ensure that a connection can be established via the external SIM card, as previously mentioned, the valid APN settings must be saved in the myDatalogEASY IoT GW ATEX and the chargeable "Activation code VPN SIM (300539)" feature must be released.*

ACTIVE, STARTED, NOT ACTIVE, WAKEUPABLE, RETRY, FAILED

see "Procedure in case of connection aborts" on page 42

The current status can be read out at any time via the "rM2M_TxGetStatus()" function.

Chapter 7 Storage, delivery and transport

7.1 Inspection of incoming deliveries

Check the shipment immediately upon receipt to ensure it is complete and intact. Immediately report any discovered transport damages to the delivering carrier. Also notify Microtronics Engineering GmbHin writing about this without delay. Report any incompleteness of the delivery to the responsible representative or directly to the company headquarters of the manufacturer within two weeks (see "Contact information" on page 293).

Note: *Any claims received thereafter will not be accepted.*

7.2 Scope of supply

Note: *The power supply unit required for operation (see "Power supply units" on page 281) and the antenna (see "Antennas" on page 281) are not part of the standard scope of delivery and must be ordered separately.*

The standard scope of delivery of the myDatalogEASY IoT GW ATEX includes:

- myDatalogEASY IoT GW ATEX base unit
- 4x hexagon socket screw M6x30
- Housing cover
- Gas protection set for myDatalogEASY IoT series (301414)
- MDN Magnet (206.803)

How-To-Video: Unpacking the myDatalogEASY IoT GW ATEX

Additional accessories such as assembly sets, antennas, power supply units, charger, etc., depending on the order. Please check these against the delivery slip.

7.3 Storage

The following storage conditions must be observed:

myDatalogEASY IoT GW ATEX	Storage temperature	-20...+50°C
	Humidity	15...90%rH
PSU713 BP (300526)	Operating temperature	-20...+50°C
	Storage temperature	+20...+25°C
PSU DC (300529)	Operating temperature	-20...+60°C
	Storage temperature	0... +35°C

Note: *The table above only contains the storage conditions for the energy sources used most frequently for the myDatalogEASY IoT GW ATEX . Please consult the appropriate factsheet for information about the storage conditions of other power supply units.*

Important note: *Remove the power supply unit from the myDatalogEASY IoT GW ATEX prior to storage.*

Store the measurement technology so that it is protected against corrosive or organic solvent vapours, radioactive emissions as well as strong electromagnetic radiation.

7.4 Transport

Protect the myDatalogEASY IoT GW ATEX against heavy shocks, bumps, impacts or vibrations. The original packaging must always be used for transport.

7.4.1 Transporting power supply units

WARNING:
With the exception of the PSU DC (300529), the power supply units required to operate the myDatalogEASY IoT GW ATEX are classified as hazardous goods due to the installed battery packs for which the following conditions must be observed during transport.

The guidelines that must be observed when transporting hazardous goods are dependent on the selected transport route and are designed as follows:

- Transport by road: ADR Directive
- Transport by air: IATA guideline
- Transport by rail: RID guideline
- Transport by ship: IMDG guideline

The following parties must observe these guidelines:

- Shipping and packaging company
- Haulage contractors
- Air carriers and ground handling service providers (only if the IATA guideline is applied)
- Security personnel (particularly if the IATA guideline is applied)

When it comes to transporting hazardous goods, the most important tasks of the shipping and packaging company are:

- Classification / identification
- Packaging
- Marking and labelling (e.g. transport stickers)
- Documentation (e.g. ADR transport documents or Dangerous Goods Declaration)

The energy stores of the power supply unit are lithium batteries. The following points must therefore be observed in conjunction with transporting lithium batteries:

- Type of lithium battery
  - Lithium ion battery
  - Lithium metal battery
- Battery energy content
  - If the energy content is within the "exempt" level, this will make the transportation process easier.
  - If the energy content is above the "exempt" level, the battery is classed as a "complete" hazardous material in accordance with the relevant guideline.
- Scope of the delivered package
  - Battery packed individually
  - Battery packed with or in the equipment

Other matters that must be taken into consideration during transport preparation and processing:

- Net weight per package
- Involved air carrier (only if the IATA guideline is applied)
- Destination country (only if the IATA guideline is applied)

> ***Important note:*** *All of these points must be observed when compiling each delivery (including the interdependencies).*

The "shipping and packaging company" can refer to the information provided in the following table to classify the power supply units and subsequently determine which transport guideline to apply.

Title of the PSU	Order number	Type of lithium battery	Energy [Wh]	Lithium content [g]	UN number
PSU713 BP	300526	Lithium metal		6,6g	UN3090

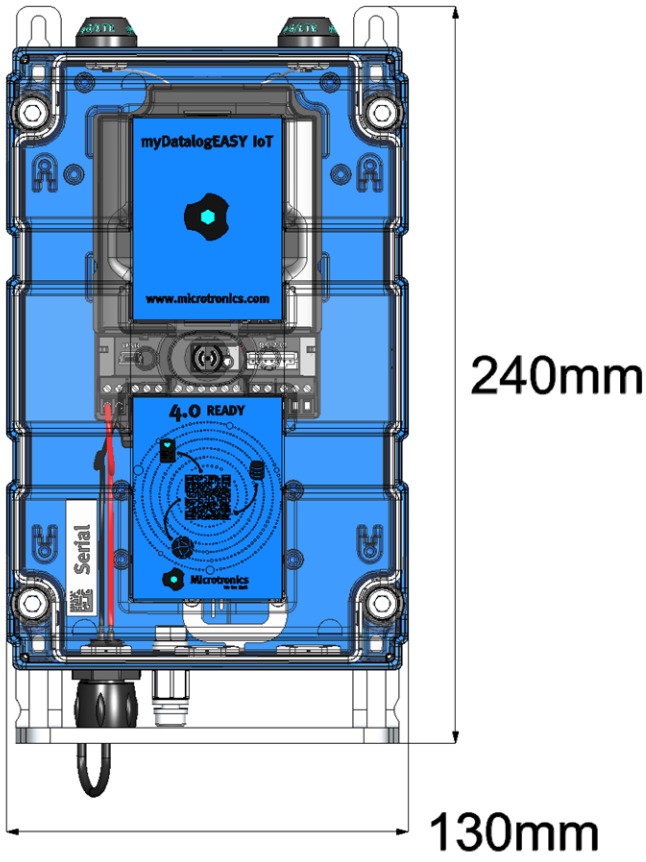
7.5 Return

Every return must be accompanied by a fully field-out return form. This return form is available in the service area of the myDatanet server. An RMA number is mandatory for any returns and can be obtained from the Support & Service Centre (see "Contact information" on page 293). The return shipment of the myDatalogEASY IoT GW ATEX must occur in the original packaging and with freight and insurance paid to Microtronics Engineering GmbH (see "Contact information" on page 293). Insufficiently cleared return shipments will otherwise not be accepted!

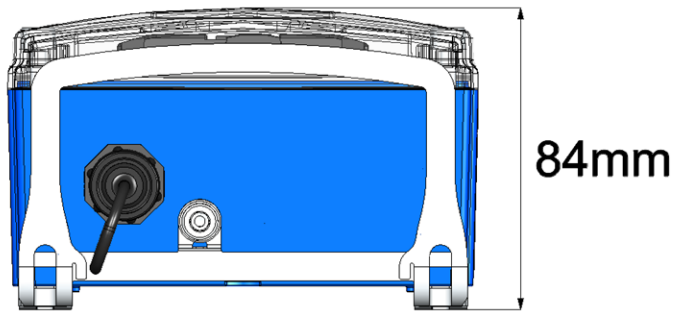
Chapter 8 Installation

Important note: *To prevent any damage to the device, the work described in this section of the instructions must only be performed by qualified personnel.*

8.1 Dimensions



Dimensions: width and height

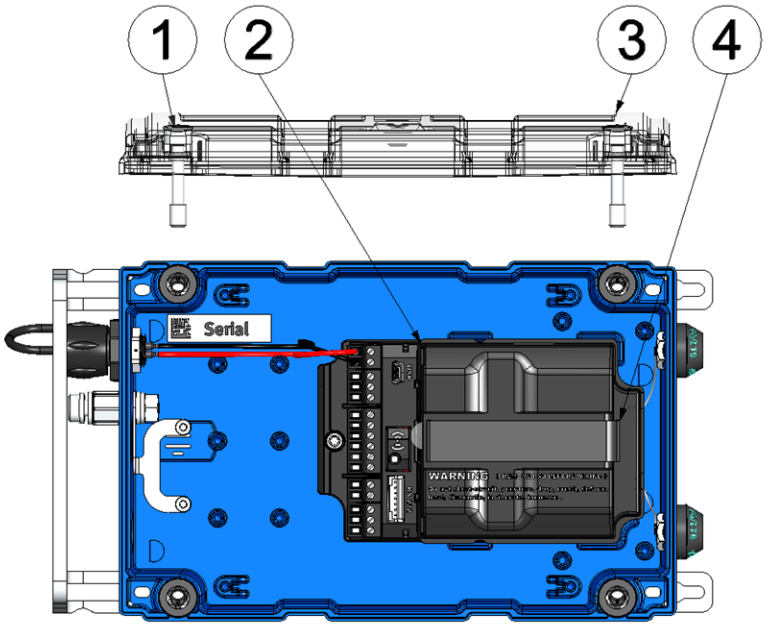


Dimensions: depth

8.2 Inserting/replacing the SIM card

Note: *The chargeable feature "Activation code VPN SIM (300539)" must be released to be able to use the SIM slot.*

How-To-Video: Inserting the SIM card



Opening the myDatalogEASY IoT GW ATEX

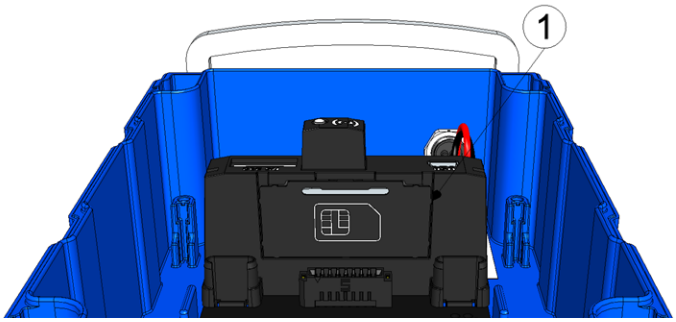
1 Hexagon socket screw M6x30	**3** Housing cover
2 Power supply unit	**4** Strap to remove the power supply unit

1. Remove the four screws that secure the housing cover. Now open the myDatalogEASY IoT GW ATEX .

   Important note: *In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing cover is open.*

2. Remove the power supply unit from the myDatalogEASY IoT GW ATEX . Use the strap provided to remove the power supply unit.

3. Remove the SIM slot cover.



Opening the SIM slot cover

1 SIM slot cover	

4. Insert the SIM as illustrated in Figure B on the circuit board of the myDatalogEASY IoT GW ATEX .

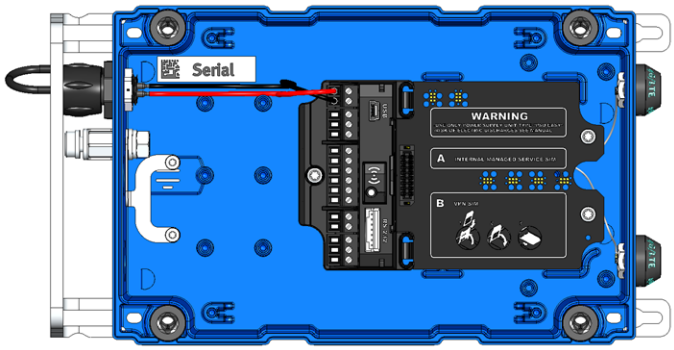


Figure on the circuit board of the myDatalogEASY IoT GW ATEX

The following step is only necessary if you want to test the connection to the myDatanet server afterwards.

5. Connect the antenna (see "Connecting the mobile network antennas" on page 74). The antenna is not included in the scope of delivery and must be ordered separately.
6. Reinsert the cover of the SIM slot and the power supply unit.

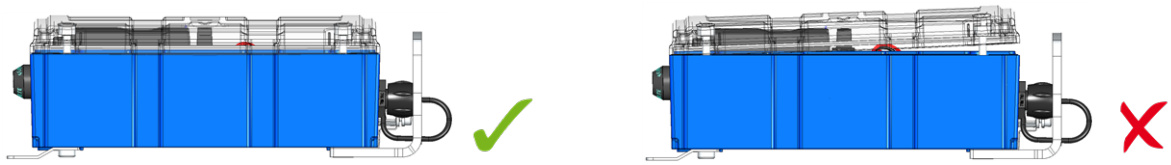
The following step is not mandatory.

7. Check whether the connection to the myDatanet server has worked correctly (see "Testing communication with the device" on page 81).
8. Close the housing cover. The best option is to tighten the four screws crosswise (torque: max. 1Nm) so that the housing cover is positioned evenly.

> ***Important note:*** *Ensure that the seals are clean and intact before closing the housing cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.*

9. Check that the housing cover is positioned correctly on all sides and that no foreign materials have been trapped between the housing and housing cover.

   Important note: *The manufacturer is not liable for any damage that is caused by housing covers that are not closed correctly.*



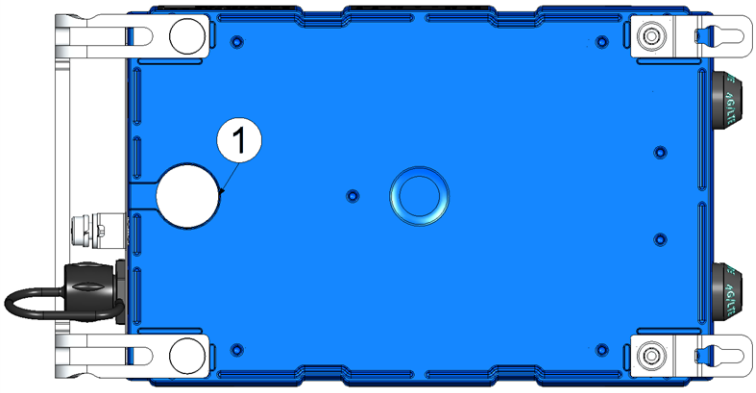
8.3 Sealing the pressure compensation

If the myDatalogEASY IoT GW ATEX is used in slightly corrosive environments, the pressure compensation must be sealed using the sticker included in the Gas protection set for myDatalogEASY IoT series (301414). The Gas protection set for myDatalogEASY IoT series is included in the scope of delivery of the myDatalogEASY IoT GW ATEX .

Important note:

- *The sticker does not protect against high concentrations of highly corrosive gases.*
- *When sealing the pressure compensation, the silica gel pouch must also be inserted in the device of the myDatalogEASY IoT GW ATEX .*

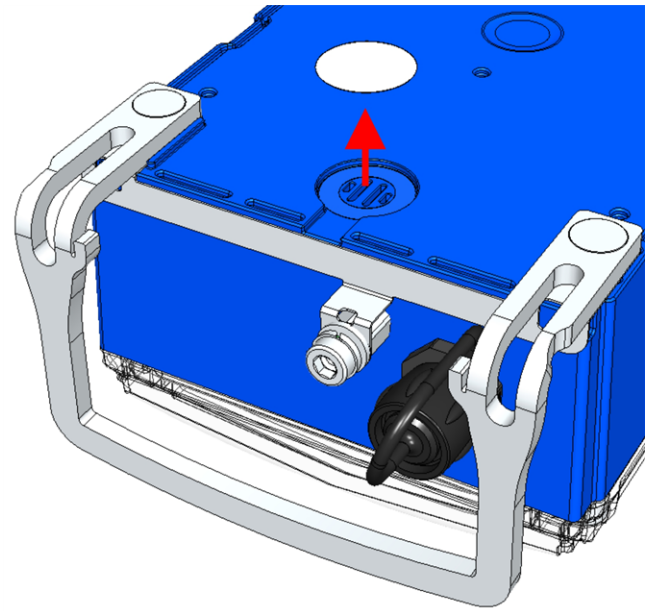
How-To-Video: Sealing the pressure compensation



Rear of the myDatalogEASY IoT GW ATEX

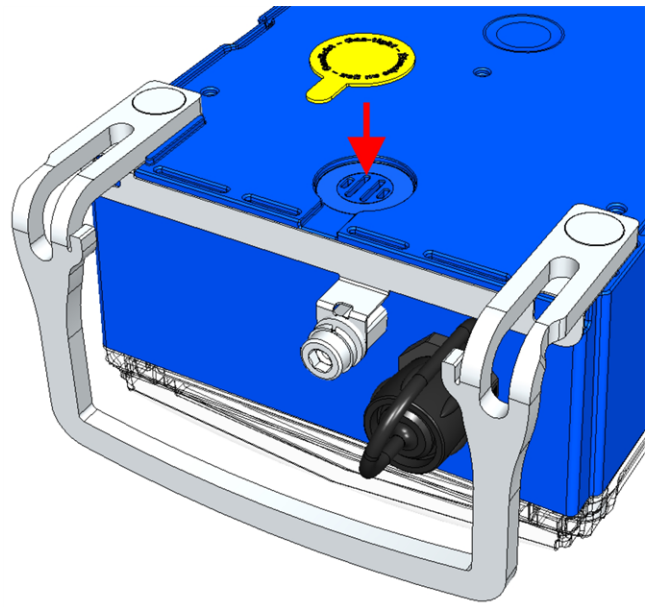
1	Pressure compensation

1. Remove the pressure compensation membrane.



Removing the membrane

2. Clean the adhesive surface with the cleaning cloth included in the Gas protection set for myDatalogEASY IoT series (301414).
3. Seal the vent with the sticker included in the Gas protection set for myDatalogEASY IoT series .



Sealing the vent

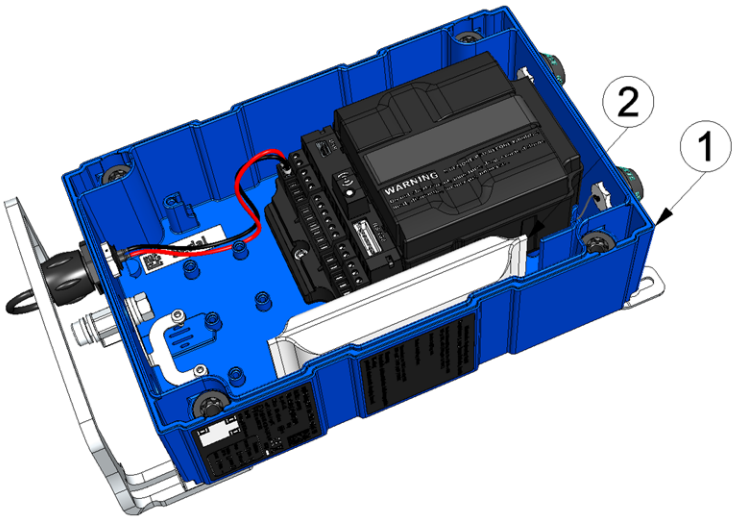
4. Remove the four screws that secure the housing lid. Then open the myDatalogEASY IoT GW ATEX .

> ***Important note:*** *In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing cover is open.*

5. Insert the silica gel pouch in the myDatalogEASY IoT GW ATEX .

   Important note: *Ensure that the pouch is inserted properly so that it is not pinched between the base part and the housing lid when you later close the housing, thus endangering the tightness of the device.*

How-To-Video: Replacing the PSU and the silica gel pouch



Inserting the silica gel pouch

1 myDatalogEASY IoT GW ATEX base part	**2** Silica gel pouch

Proceed as follows during this process:

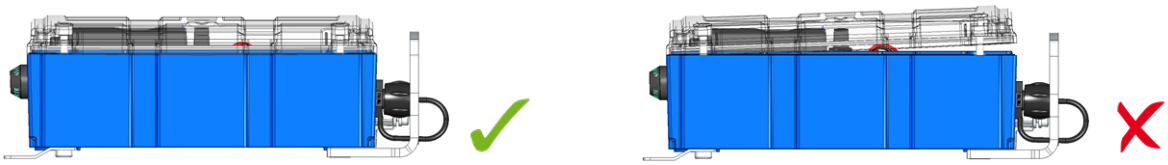
1. Hold the pouch in a way that the longer side is pointing downwards.
2. Shake the bag to distribute the silica gel evenly along this longer side of the bag.
3. Fold over the top empty area of the bag to create a more compact bulging bag.
4. Insert the pouch with this folded side down in the myDatalogEASY IoT GW ATEX .

6. Close the housing lid. The best option is to tighten the four screws crosswise (torque: max. 1Nm) so that the housing cover is positioned evenly.

   Important note: *Ensure that the seals are clean and intact before closing the housing cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.*

7. Check that the housing cover is positioned correctly on all sides and that no foreign materials have been trapped between the housing and housing cover.

   Important note: *The manufacturer is not liable for any damage that is caused by housing covers that are not closed correctly.*



8.4 Installing the myDatalogEASY IoT GW ATEX

Important note:

- *Ensure installation is completed correctly.*
- *Comply with existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the devices.*
- *The myDatalogEASY IoT GW ATEX must not be operated in the field with the lid open.*
- *The pressure compensation must be protected against contamination.*
- *The myDatalogEASY IoT GW ATEX is not approved for use in closed channels.*
- *When using metal mounting components, these must be integrated into the grounding concept in a suitable manner.*

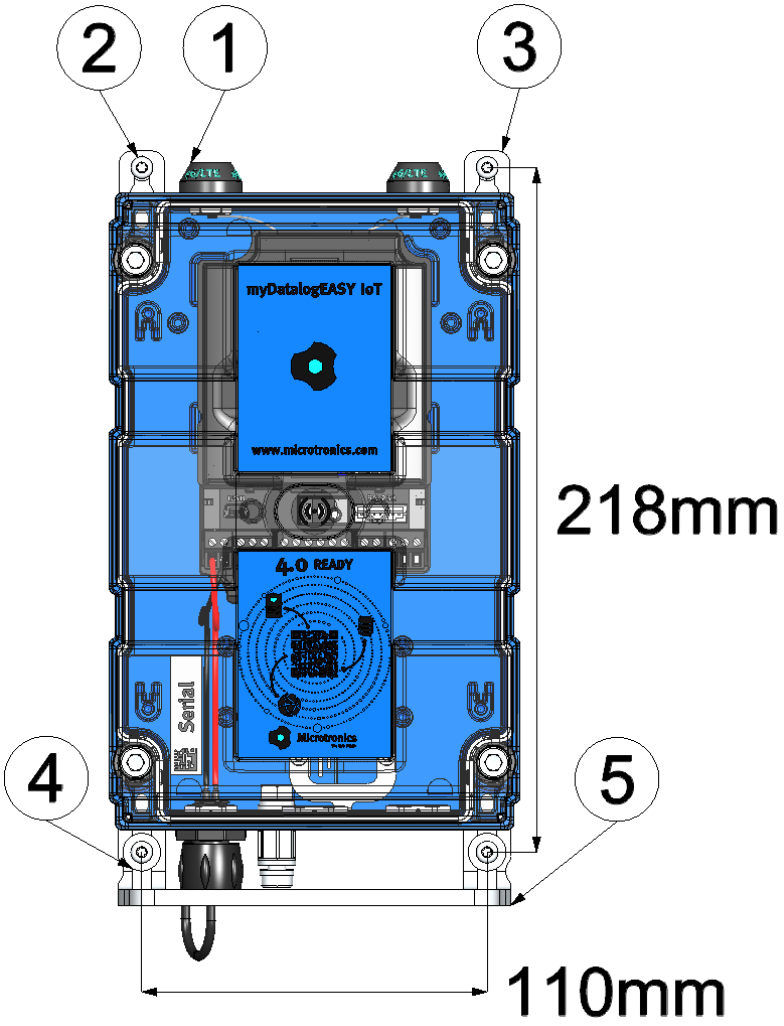
The installation site must be selected according to specific criteria. The following conditions must be avoided in any case:

- Direct sunlight
- Direct weather exposure (rain, snow, etc.)
- Objects that radiate intense heat (maximum ambient temperature: -20...+50°C)
- Objects with a strong electromagnetic field (frequency converter or similar)
- Corrosive chemicals or gases
- Mechanical impacts
- Direct installation on paths or roads
- Vibrations
- Radioactive emissions

Note: *Leave sufficient space at the upper end to install the antenna. The space required depends on the antenna used. Approx. 15 cm of space must be left beneath the device for the cable connections. Further information regarding the installation dimensions can be found in the relevant sub-chapter.*

8.4.1 Wall mounting

The assembly loops required for wall mounting are already attached to the myDatalogEASY IoT GW ATEX upon delivery.



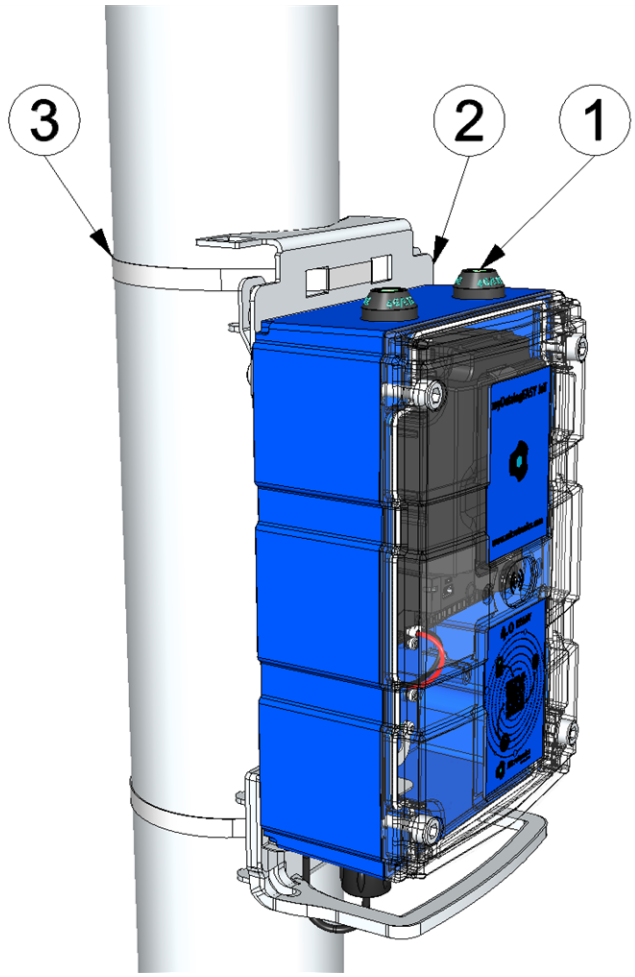
Wall mounting

1	myDatalogEASY IoT GW ATEX	**4**	Lining disc (optional, min. diameter: 12 mm)
2	Screw (max. diameter: 4 mm)	**5**	Metal bracket to protect the connectors
3	Assembly loop		

1. Drill the four holes for mounting in accordance with the dimensions specified in figure "Wall mounting" on page 62. The diameters are determined by the screws that are being used and any wall plugs that may be required.
2. If necessary, insert a wall plug into each of the four drilled holes before you screw the myDatalogEASY IoT GW ATEX to the wall with the attached assembly loops (see "Wall mounting" on page 62). Use suitable lining discs, if required. This specifically applies to the two screws that are inserted through the slot of the metal holder to protect the connections.

8.4.2 Pipe mounting

For the installation on a pipe the optional accessory "Pipe mounting EASY IoT 30-250 mm (301191)" is required. If the myDatalogEASY IoT GW ATEX and the Pipe mounting EASY IoT 30-250 mm are ordered together, the mounting loops included in the standard scope of delivery are replaced by those for the installation on a pipe and are mounted on the myDatalogEASY IoT GW ATEX prior to shipment



Pipe mounting

1 myDatalogEASY IoT GW ATEX	**3** Tightening strap (for pipe diameter 30-250mm)
2 Loops for pipe mounting (for pipe diameter 30-250mm)	

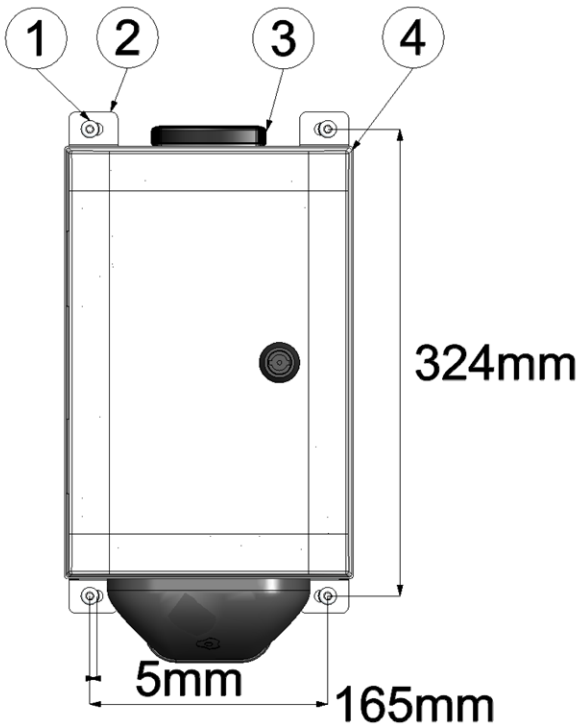
1. Thread the tightening straps (3) through the appropriate openings in the mounting loops (2).
2. Position the myDatalogEASY IoT GW ATEX with attached mounting loops (2) on the pipe and use the provided tightening straps (3) to fasten the myDatalogEASY IoT GW ATEX .

8.4.3 Outdoor installation

When installing the myDatalogEASY IoT GW ATEX outdoors, it needs to be protected from direct sunlight and direct weather exposure (rain, snow, ...). To do so, the "Housing ATEX 300x200x150mm for outdoor mounting (301238)" can be used. The housing for outdoor installation already includes a multiband antenna suitable for all variants of the myDatalogEASY IoT GW ATEX .

8.4.3.1 Attaching the housing for outdoor installation to a wall

To attach the "Housing ATEX 300x200x150mm for outdoor mounting (301238)" to a wall, the "Wall mounting set for housing 300x200x150mm(301185)" is required as an additional accessory. If the myDatalogEASY IoT GW ATEX , the Housing ATEX 300x200x150mm for outdoor mounting and the Wall mounting set for housing 300x200x150mm are ordered together, the myDatalogEASY IoT GW ATEX is installed in the housing for outdoor installation (incl. connection of antenna cables) and the brackets for installation on a wall are mounted on the housing prior to shipment.



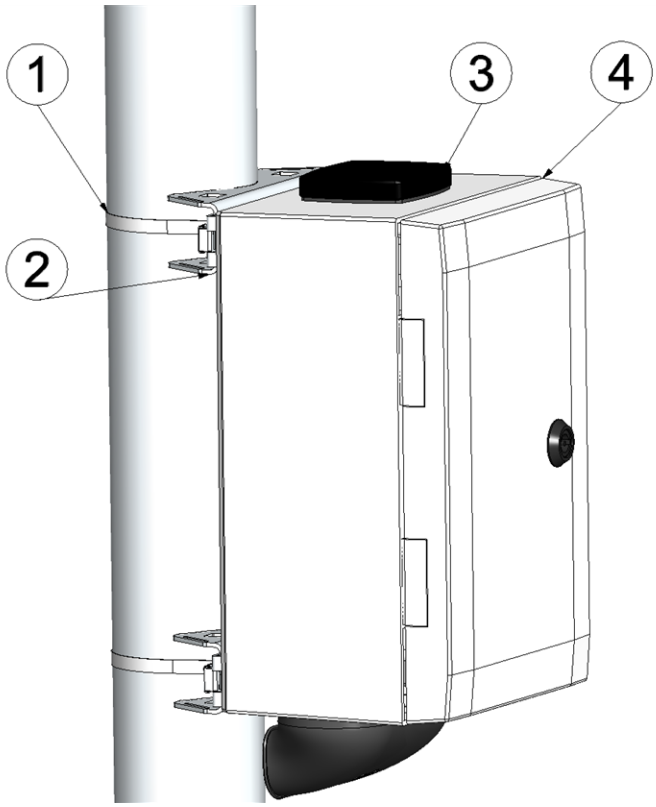
Wall mounting

1 Screw (max. diameter: 8mm)	**3** Multiband antenna
2 Mounting bracket	**4** Housing ATEX 300x200x150mm for outdoor mounting

1. Drill the four holes for mounting in accordance with the dimensions specified in figure "Wall mounting" on page 64. The diameters are determined by the screws that are being used and any wall plugs that may be required.
2. If necessary, insert a wall plug into each of the four drilled holes before you screw the Housing ATEX 300x200x150mm for outdoor mounting with the installed mounting brackets (2) to the wall (see "Wall mounting" on page 64).

8.4.3.2 Attaching the housing for outdoor installation to a pipe

To attach the "Housing ATEX 300x200x150mm for outdoor mounting (301238)" to a pipe, the " Pipe mounting set for housing 300x200x150mm (301184)" is required as an additional accessory . If the myDatalogEASY IoT GW ATEX , the Housing ATEX 300x200x150mm for outdoor mounting and the Pipe mounting set for housing 300x200x150mm are ordered together, the myDatalogEASY IoT GW ATEX is installed in the housing for outdoor installation (incl. connection of antenna cables) and the brackets for the installation on a pipe are mounted on the housing prior to shipment.



Pipe mounting

1	Tightening strap (for pipe diameter 60-300mm)	**3**	Multiband antenna
2	Bracket for pipe mounting (for pipe diameter 60-300mm)	**4**	Housing ATEX 300x200x150mm for outdoor mounting

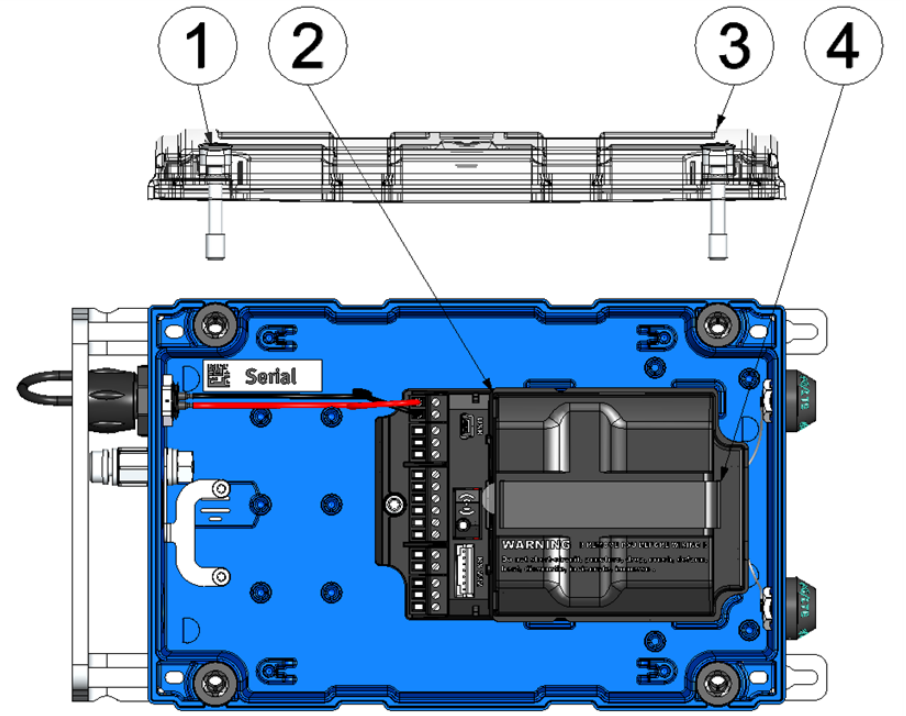
1. Position the Housing ATEX 300x200x150mm for outdoor mounting with the installed mounting brackets (2) on the pipe and use the tightening straps (1) icluded in the scope of delivery to attach the Housing ATEX 300x200x150mm for outdoor mounting .

8.5 Safety instructions for cabling

Important note: *To avoid any damage, always switch off the voltage supply to the device when performing electrical connections.*

When connections are made to the myDatalogEASY IoT GW ATEX , the following warnings and information must be observed, in addition to the warnings and information found in the individual chapters on the installation. Further safety information is included in "Safety instructions" on page 25.

Remove the power supply unit from the device before completing any wiring work.



Removing the power supply unit

1 Hexagon socket screw M6x30	**3** Housing cover
2 Power supply unit	**4** Strap to remove the power supply unit

8.5.1 Information on preventing electrostatic discharges (ESD)

Important note: *Maintenance procedures that do not require the device to be connected to the power supply should only be performed once the device has been disconnected from the mains power supply to minimise hazards and ESD risks.*

The sensitive electronic components inside the device can be damaged by static electricity, which can impair the device performance or even cause the device to fail. The manufacturer recommends the following steps to prevent any damage to the device caused by electrostatic discharges:

- Discharge any static electricity present on your body before handling the electronic components of the device (such as circuit boards and components attached thereto). To do this, you can touch a grounded metallic surface such as the housing frame of a device or a metal pipe.
- Avoid any unnecessary movements to prevent the build-up of static charges.
- Use antistatic containers or packaging to transport components that are sensitive to static.
- Wear an antistatic wristband that is grounded via a cable to discharge your body and keep it free of static electricity.
- Only touch components that are sensitive to electric charges in an antistatic working area. If possible, use antistatic mats and work pads.

8.6 Electrical installation

Important note: *Only qualified personnel should undertake the installation described in this chapter of the operating instructions to avoid any damage to the device.*

8.6.1 Supply concepts

The following table gives an overview of the possible supply concepts including the respective advantages and disadvantages:

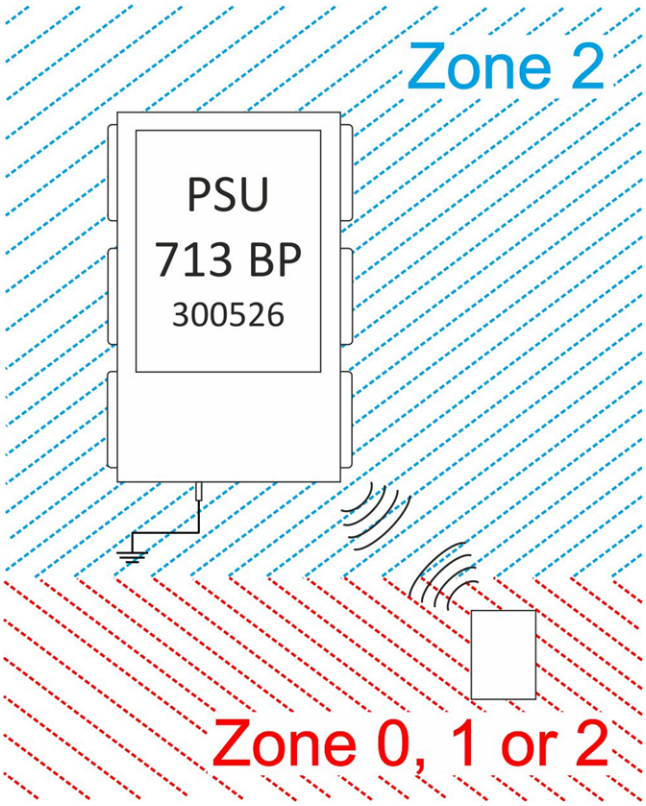
Supply concept	Advantages	Disadvantages
Supply by primary battery	• Minimal installation effort • No energy source required at the installation site	• Requires a cyclic replacement of the power supply unit
Supply by PV module	• Self-sufficient operation • No energy source required at the installation site	• Due to the energy supply using a PV module, there is a dependency on insolation
Mains operation	• No cyclic passing by the measurement site needed • No dependency on insolation	• Energy source required at the installation site • High installation effort

8.6.1.1 Powered by primary battery

This supply concept is distinguished by the least installation effort. It is limited to installing the device (see "Installing the myDatalogEASY IoT GW ATEX " on page 61), connecting the earthing connection of the myDatalogEASY IoT GW ATEX with the ground potential (see "Connection of the power supply" on page 71) and earthing all metallic mounting fixtures in a suitable manner. The disadvantage is that the power supply unit equipped with a primary battery has to be cyclically (depending on the transmission and measurement interval) changed.

The following components are required to implement the the supply concept:

- PSU713 BP (300526)



Schematic diagram: supply via primary battery

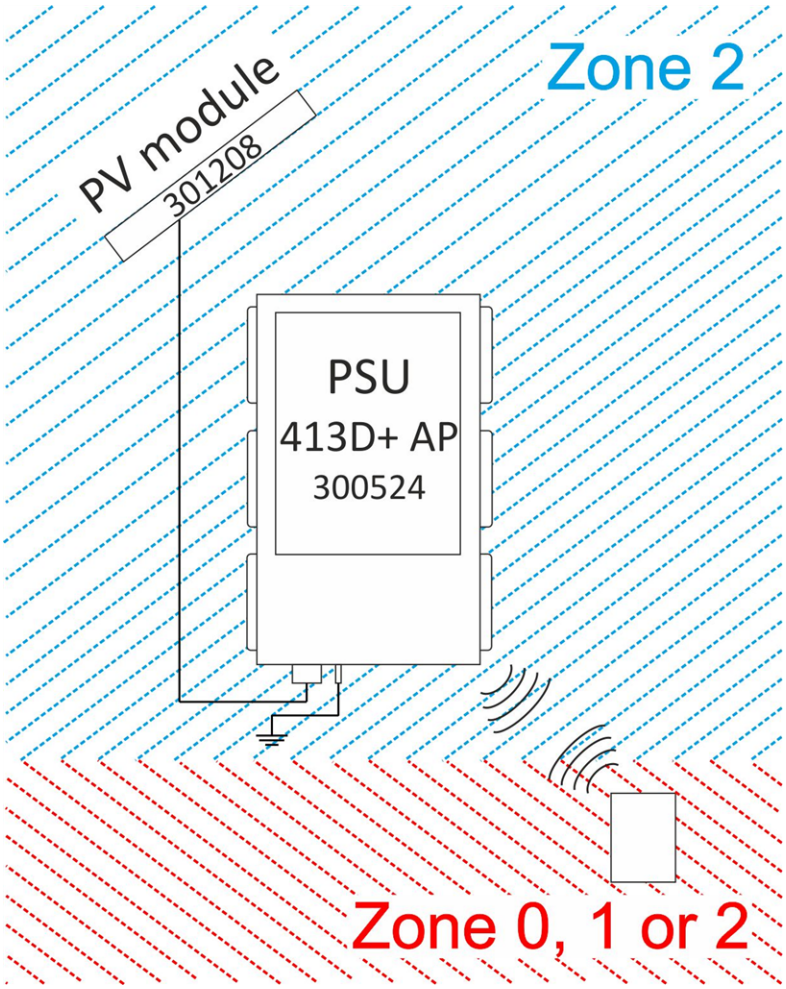
The myDatalogEASY IoT GW ATEX may only be operated in areas with a zone 2 explosive atmosphere. Thanks to the communication via Bluetooth (Bluetooth Low Energy), there is no electric connection between the myDatalogEASY IoT GW ATEX and the sensors. Therefore the sensors may be operated in areas with a zone 0, zone 1 or zone 2 explosive atmosphere. The sensors must of course be approved for the relevant zone.

8.6.1.2 Powered by PV module

This supply concept enables a self-sustaining operation of the myDatalogEASY IoT GW ATEX . I.e. no energy source has to be available at the installation site. When installing the measuring device, it is important to note that all metallic mounting fixtures have to be grounded in a suitable manner and that the earthing connection of the myDatalogEASY IoT GW ATEX has to be connected to the ground potential (see "Connection of the power supply" on page 71).

The following components are required to implement the supply concept:

- PV module 30W for ATEX myDatalogEASY IoT series (301208)
- PSU413D+ AP (300524)



Schematic diagram: supply via PV module

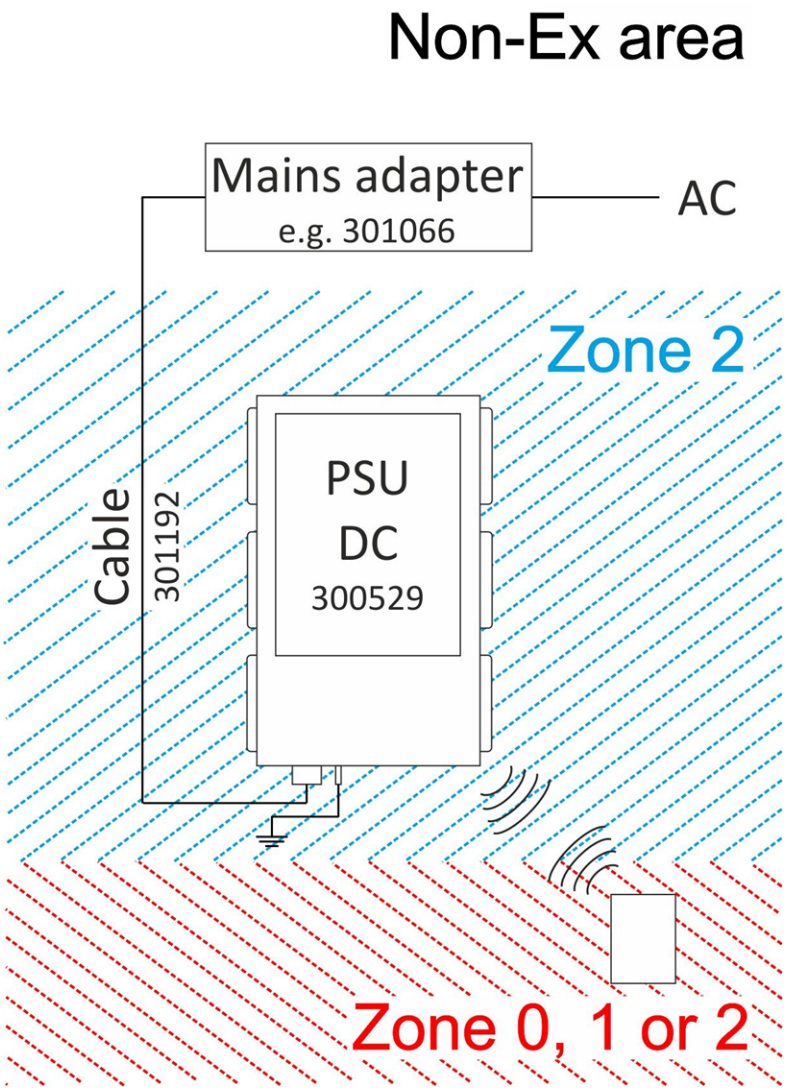
The myDatalogEASY IoT GW ATEX and the "PV module 30W for ATEX myDatalogEASY IoT series " may only be operated in areas with a zone 2 explosive atmosphere. Thanks to the communication via Bluetooth (Bluetooth Low Energy), there is no electric connection between the myDatalogEASY IoT GW ATEX and the sensors. Therefore the sensors may be operated in areas with a zone 0, zone 1 or zone 2 explosive atmosphere. The sensors must of course be approved for the relevant zone.

8.6.1.3 Powered by mains adapter

This supply concept is particlularly suitable if an energy source is already available at the installation site. When installing measuring device, it is important to note that all metallic mounting fixtures have to be included in the earthing concept of the existing installation in a suitable manner and that the earthing connection of the myDatalogEASY IoT GW ATEX is connected to the ground potential of the existing installation (see "Connection of the power supply" on page 71).

The following components are required to implement the supply concept:

- Power supply 24V 0,63A for top-hat rail mounting (301066)
- Connection cable 2-pins for supply 5m (301192)
- PSU DC (300529)



Schematic diagram: mains operation

The myDatalogEASY IoT GW ATEX may only be operated in areas with a zone 2 explosive atmosphere. Using the "Connection cable 2-pins for supply 5m ", the myDatalogEASY IoT GW ATEX is connected to the mains adapter (e.g. "Power supply 24V 0,63A for top-hat rail mounting "), which is located outside the area with explosive atmosphere.

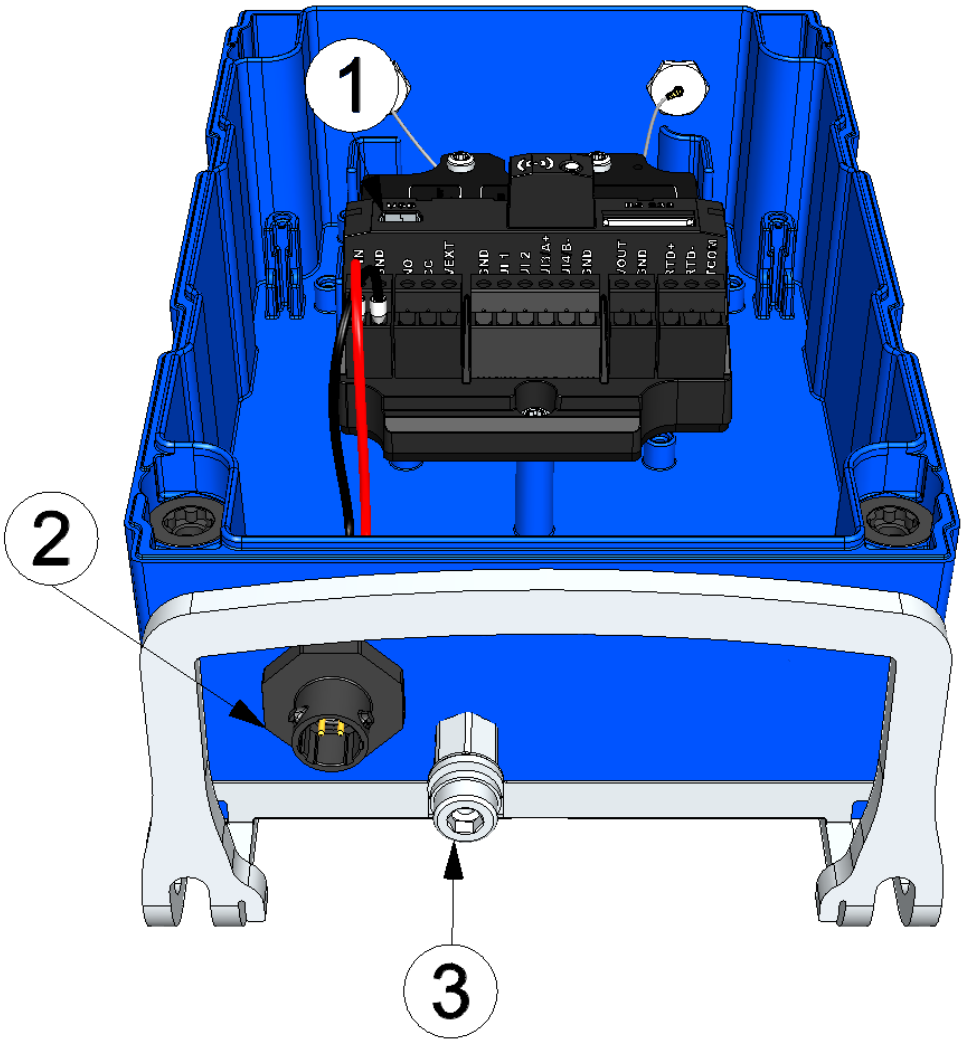
> ***Important note:*** *The connection point between the "Connection cable 2-pins for supply 5m " and the mains adapter always has to be outside the area with a zone 2 explosive atmosphere.*

Thanks to the communication via Bluetooth (Bluetooth Low Energy), there is no electric connection between the myDatalogEASY IoT GW ATEX and the sensors. Therefore the sensors may be operated in areas with a zone 0, zone 1 or zone 2 explosive atmosphere. The sensors must of course be approved for the relevant zone.

8.6.2 Connection of the power supply

Important note:

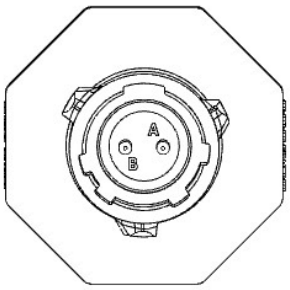
- *All wiring work must be performed in the de-energised state.*
- *Ensure installation is completed correctly.*
- *Comply with existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the instruments.*
- *Run all data and power cables so that they do not pose a trip hazard and ensure that cables do not have any sharp bends.*
- *The myDatalogEASY IoT GW ATEX must not be operated in the field with the cover open.*
- *The myDatalogEASY IoT GW ATEX cannot be operated without a power supply unit.*



Connection of the power supply (view without power supply unit)

1	Mini-B USB (only for debugging and device logic update)	**3**	Earth connection (Minimum requirement for the earth cable: 4 mm²)
2	Connector for supply or charging voltage		

Connection for the supply or charging voltage

	Connection for the supply or charging voltage (2-pin MIL connector)	Connection cable 2-pins for supply 5m (301192)	Signal	
 Connection for the supply or charging voltage	A	1	External supply or charging voltage 12...32VDC (max. 12W)	+
	B	2	Ground	-

Note: *The first two steps are only necessary if the device is already in operation and the wiring needs to be modified.*

1. Remove the four screws that secure the housing cover. Now open the myDatalogEASY IoT GW ATEX .

   Important note: *In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing cover is open.*

2. Remove the power supply unit from the myDatalogEASY IoT GW ATEX . Use the strap provided to remove the power supply unit.

3. Connect the ground connection of the myDatalogEASY IoT GW ATEX to the ground potential of the system.

4. If you would like to use an external supply or charging voltage, you should connect the corresponding cable with the supply or charging voltage connection in the de-energised state. Use the Connection cable 2-pins for supply 5m (301192) to connect your power source to the connection for the supply or charging voltage. Further accessories are detailed in the chapter "Spare parts and accessories" on page 281.

   Important note: *The connection point between the "Connection cable 2-pins for supply 5m " (301192) and the power source must always be outside the area with a zone 2 explosive atmosphere.*

   Important note: *Use suitable means to protect the connection point between the "Connection cable 2-pins for supply 5m " (301192) and power source against the penetration of moisture and dust. You can use the Clamping tube(300256) for this purpose.*

5. Affix corresponding protective caps to all of the connectors that are not required.

   Important note: *All unused connectors of the myDatalogEASY IoT GW ATEX must be sealed watertightly using the protective caps supplied. Otherwise the degree of protection for the entire device is not guaranteed and the manufacturer's warranty is void.*

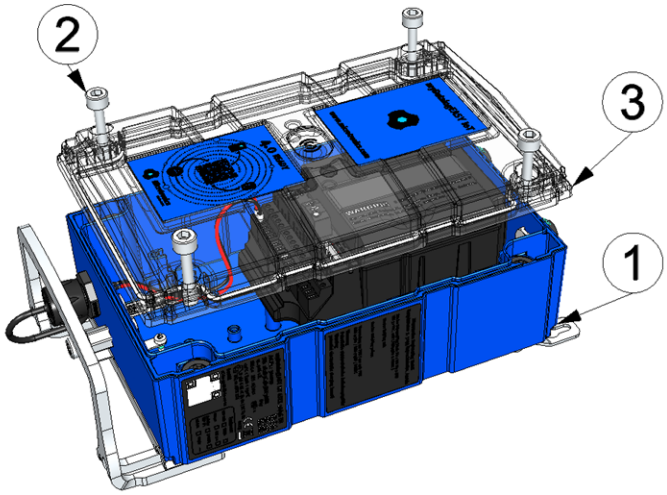
6. Connect the antenna (see "Connecting the mobile network antennas" on page 74). The antenna is not included in the scope of delivery and must be ordered separately.

7. Insert the power supply unit.

The following step is not mandatory.

8. Check whether the connection to the myDatanet has worked correctly (see "Testing communication with the device" on page 81).
9. Close the housing cover. The best option is to tighten the four screws crosswise (torque: max. 1Nm) so that the housing cover is positioned evenly.

   Important note: *Ensure that the seals are clean and intact before closing the housing cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.*

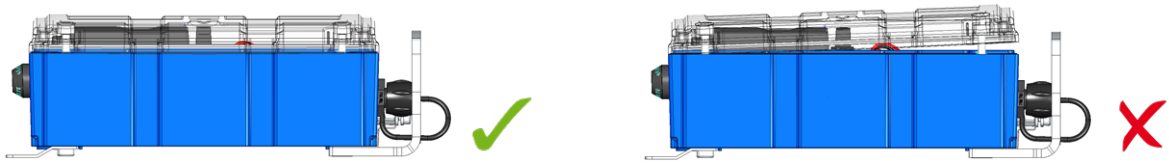


Closing the housing cover

1 myDatalogEASY IoT GW ATEX base unit	**3** Housing cover
2 Hexagon socket screw M6x30	

10. Check that the housing cover is positioned correctly on all sides and that no foreign materials have been trapped between the housing and housing cover.

    Important note: *The manufacturer is not liable for any damage that is caused by housing covers that are not closed correctly.*



The following step is only necessary if you are using an external supply or charging voltage.

11. Now switch on the external supply or charging voltage.

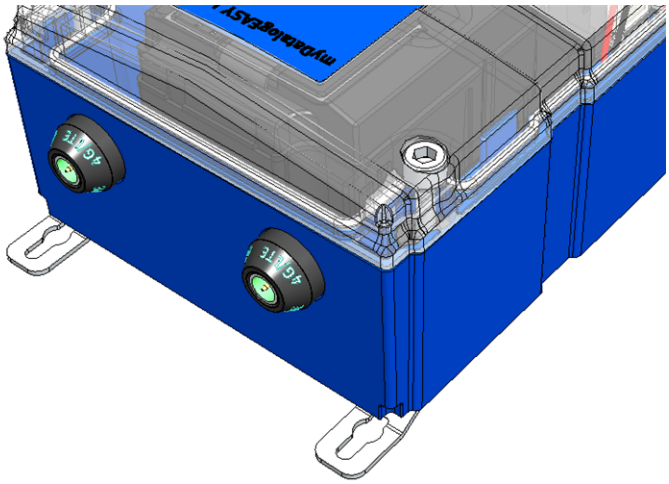
    Note: *If you are using a power supply unit without an integrated energy store, the external supply or charging voltage must be switched on before the optional step during which the connection to the server is tested.*

8.6.3 Connecting the mobile network antennas

> ***Important note:*** *To ensure the correct functionality, only use antennas that are supplied by the manufacturer.*

Depending on the variant, the myDatalogEASY IoT GW ATEX disposes of up to 2 FME-M antenna connections. The configuration is marked on each antenna connector. A list of available antennas can be found in chapter "Antennas" on page 281.

> ***Note:*** *Make sure that the selected antenna is suitable for the radio technology (e.g. 2G/3G) installed in the myDatalogEASY IoT GW ATEX and for the intended region (e.g. Europe or the US).*



Antenna connectors of the myDatalogEASY IoT GW ATEX (e.g. myDatalogEASY IoT 2G/4G EU)

The standard antenna (Portable antenna multi band FME-F , 206.826) is attached directly to the antenna connector (see "Overview" on page 28) of the myDatalogEASY IoT GW ATEX . In the event of a low radio signal strength, you can use the Dome antenna multi band FME-F 3m (301211) .

If the distance between the antenna position and the myDatalogEASY IoT GW ATEX is too great, you can use a 5m Extension cable for antenna FME-F/FME-M 5m (206.805).

1. Ensure that the myDatalogEASY IoT GW ATEX is de-energised.
2. If you need an antenna extension, connect it to the antenna first.
3. Connect the antenna extension or antenna directly to the antenna connector of the myDatalogEASY IoT GW ATEX (see "Overview" on page 28).

   > ***Important note:*** *Do not apply too much force when tightening the antenna. Do not use any tools to tighten the antenna or antenna extension; only tighten it manually.*

4. Switch the voltage supply of the myDatalogEASY IoT GW ATEX back on.

The following step is not mandatory.

5. Check whether the connection to the myDatanet server has worked correctly (see "Testing communication with the device" on page 81).

8.6.4 Technical details about the USB interface

The connection to a PC is established via the USB slave interface. It is only designated for the communication with the web-based development environment rapidM2M Studio or the DeviceConfig configuration program. It is not possible to access the USB interface via the device logic. A detailed description of the rapidM2M Studio web-based development environment is provided in chapter "rapidM2M Studio " on page 125. Explanations regarding the functionality of the DeviceConfig configuration program is provided in chapter "DeviceConfig " on page 91.

Access to the web-based development environment rapidM2M Studio is included in the Microtronics Partner Program, for which you can register free of charge at the following address:

https://partner.microtronics.com

The DeviceConfig configuration program can be downloaded free of charge from the following website:

www.microtronics.com/deviceconfig

Important note: *If the antenna of the device is earthed or connected to the ground potential of another object (e.g. installation on a control cabinet), remove the antennas before you connect the device with the USB interface of a PC. Otherwise, this can cause a potential displacement between the ground of the antenna and the ground of the PC, which could damage the USB interface of the device.*

8.6.5 Technical details about the Bluetooth Low Energy interface

Via the Bluetooth Low Energy interface of the myDatalogEASY IoT GW ATEX , a connection to sensors and devices with a compatible interface (Bluetooth Low Energy 5.0) can be established. The Device Logic enables access to the Bluetooth interface, whereby the implementation of the specific protocol for communication with the corresponding sensor or device is render possible.

In addition, the connection to a PC or a Bluetooth Low Energy (5.0) compatible Smartphone can be established via the Bluetooth Low Energy interface. It is only designated for the communication with the DeviceConfig configuration program or the "tbd" smartphone app. A detailed description of the DeviceConfig configuration program is provided in chapter "DeviceConfig " on page 91. It can be downloaded free of charge from the following website:

www.microtronics.com/deviceconfig

Chapter ""tbd" smartphone app" on page 115 provides a detailed explanation of the "tbd" smartphone app. It is available for Android and iOS devices and can be downloaded free of charge from "Google Play" (Android) or the Apple "App Store" (iOS).

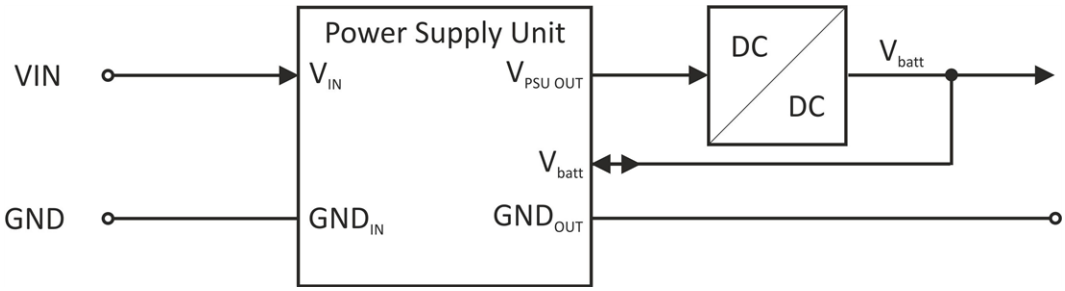
8.6.6 Technical details about energy management

The device will work until 3,4V as intended, if the myDatalogEASY IoT GW ATEX is operated without an external supply or charging voltage (V IN). The modem is deactivated from this threshold and the "UV MODEM LOCKOUT" log entry is entered in the device log. This means that the connection is disconnected if the device is in "online" mode or is logged in to the GSM network ("Interval & wakeup" mode).

The function returns the "ERROR" result if an attempt to establish the connection using the "rM2M_TxStart()" function is made once this threshold is reached. The "rM2M_TxSetMode()" function also returns the "ERROR" result if an attempt to activate "Interval & wakeup" or "online" mode is made. The Device Logic is executed as intended once this threshold is reached.

The Device Logic execution is only stopped if the internal supply voltage falls below 2,9V and the myDatalogEASY IoT GW ATEX switches to energy saving mode in which only the charge control is active. The "UV_LOCKOUT" log entry is also entered in the device log when activating energy saving mode.

8.6.7 Technical details about the energy supply



Schematic diagram of the energy supply

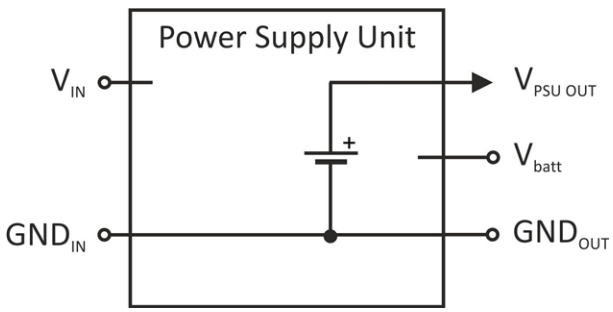
V IN	12...32VDC
Power consumption (without sensors)	typ. <1mW max. 12W
Reverse voltage protection	No [1)]

[1)] The reverse voltage protection is part of the protective circuit in the power supply units.

A selection of compatible power supplies is included in the chapter "Charging devices and power supply units" on page 282. Depending on the type, the power supply unit contains a rechargeable battery (PSU413D+ AP , PSU413D AP), a battery (PSU713 BP), or only a protective circuit (PSU DC). A list of compatible PSUs is included in the chapter "Power supply units" on page 281. An external supply or charging voltage is not required if the power supply unit is equipped with a battery.

8.6.7.1 PSU713 BP (300526)

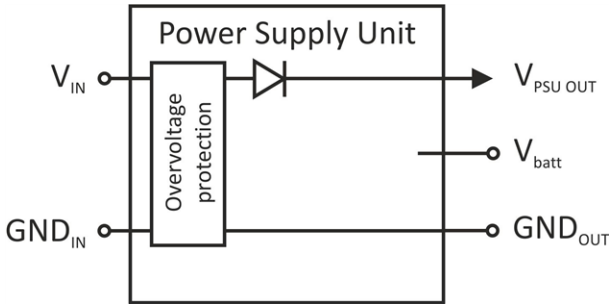
V IN	Not required
Protective circuit (V IN)	---
Capacity	13Ah 93,6Wh
Type	Li-SOCl2
Rechargeable	No
Nominal voltage	7,2V
Operating temperature	-20...+50°C
Charging temperature	---
Storage temperature	+20...+25°C



Block diagram of the PSU713 BP

8.6.7.2 PSU DC (300529)

V IN	Required
Protective circuit (V IN)	2kV overvoltage protection Reverse voltage protection
Capacity	---
Type	---
Rechargeable	no
Nominal voltage	---
Operating temperature	-20...+60°C
Charging temperature	---
Storage temperature	0... +35°C



Block diagram of the PSU DC

8.6.8 Technical details about the system time

The myDatalogEASY IoT GW ATEX is equipped with a hardware real-time clock that has its own buffer battery with an expected service life of >10 years. The system time continues to run even if the power supply unit is removed. This means that following recommissioning, valid time stamps for the measurement and log data can be generated immediately. Additionally, the system time is synchronised with the server each time a connection to the myDatanet server is established.

Chapter 9 Initial Start-Up

9.1 User information

Before you connect the myDatalogEASY IoT GW ATEX and place it into operation, you must observe and comply with the following user information!

This manual contains all information that is required for using the device.

Is intended for technically qualified personnel who have the relevant knowledge and experience in the area of measurement technology.

Read this manual carefully and completely in order to ensure the proper functioning of the myDatalogEASY IoT GW ATEX .

Contact Microtronics Engineering GmbH(see "Contact information" on page 293) if anything is unclear or if you encounter difficulties with regard to installation, connection or configuration.

9.2 Applicable documents

In addition to this operating instructions, additional instructions or technical descriptions may be required for the installation, commissioning and operation of the entire system.

These instructions are enclosed to the respective additional devices or sensors or are available for download on the Microtronics website.

9.3 General principles

The entire measurement system may only be placed into operation after completion and inspection of the installation. Study the manual thoroughly before placing into operation to prevent faulty or incorrect configuration.

Utilise the manual to familiarise yourself with the operation of the myDatalogEASY IoT GW ATEX and the input screens of the myDatanet server before you begin with the configuration.

9.4 Commissioning the system

Note: *It is recommended that the myDatalogEASY IoT GW ATEX is first placed into operation in the office before mounting the device permanently at the place of use. During this process, you should create a site for the later operation on the myDatanet server (see "Creating the site" on page 121) and determine a site configuration (including Data Descriptor and Device Logic) (see "Site configuration" on page 84). If you create the site based on an IoT application (see "myDatanet Server Manual " 805002), the Data Descriptor and Device Logic are taken from the IoT application and do not need to be defined separately. Take the opportunity to get to know the functions of the device in a stable environment. You can also use suitable test signals to simulate the sensors to establish the optimum configuration of the myDatalogEASY IoT GW ATEX prior to its actual first use. This reduces the amount of time required for on-site installation to a minimum.*

The following work should be completed in the office before you go to the future location of the device:

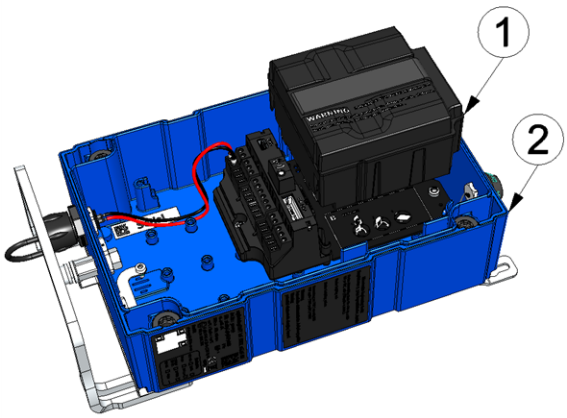
1. If necessary, create a customer on the myDatanet server (see "myDatanet Server Manual " 805002).
2. Within the selected customer, create a site/application for operation on the myDatanet server (see "Creating the site" on page 121).

   Note: *A "rapidM2M " type site or a site based on an IoT application that is compatible with the "rapidM2M " site type must be created to operate the myDatalogEASY IoT GW ATEX .*

3. Configure the created site/application according to your requirements (see "Site configuration" on page 84). If the site was not created based on an IoT application, you must determine the Data Descriptor and Device Logic via the "Control" configuration section (see "Control" on page 85).
4. Connect the antenna (see "Connecting the mobile network antennas" on page 74). The antenna is not included in the scope of delivery and must be ordered separately.
5. Establish a connection so that the configuration of site is transferred to the myDatalogEASY IoT GW ATEX . If no script has been loaded in the device yet, this can be achieved by inserting the power supply unit as described in the chapter "Replacing the power supply unit" on page 266. If a Device Logic has already been loaded in the device, execute the operations provided in the Device Logic to trigger the establishment of a connection.

   Note: *If you are using a power supply unit without an integrated energy store, the external supply or charging voltage must be connected before inserting the power supply unit. Details on this are provided in the chapter "Connection of the power supply" on page 71.*

   Note: *You can also skip this step, as a connection must also be initiated during the installation on site, which transfers the configuration to the myDatalogEASY IoT GW ATEX at the same time.*



Inserting the power supply unit

1 Power supply unit	**2** Base unit

6. Use the corresponding strap to remove the power supply unit from the myDatalogEASY IoT GW ATEX and then, if used, disconnect the cabling for the supply or charging voltage from the device when in a de-energised state, if possible.
7. Remove the antenna again.

The following work is completed directly at the location of the device:

8. Complete all of the steps detailed in the chapter "Connection of the power supply" on page 71.

9. Check whether the connection to the myDatanet server has worked correctly (see "Testing communication with the device" on page 81).

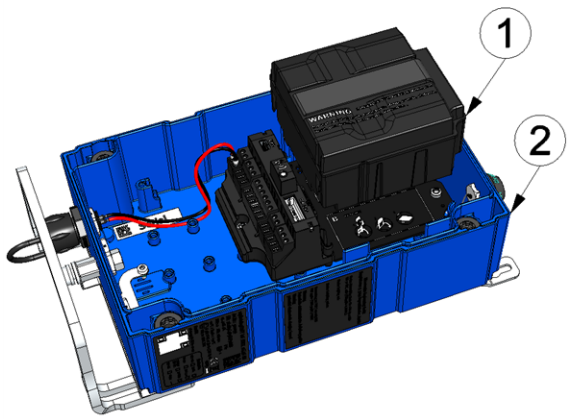
9.5 Testing communication with the device

1. Create a site for operation on the myDatanet server (see "Creating the site" on page 121).

   Note: *A "rapidM2M " type site or a site based on an IoT application that is compatible with the "rapidM2M " site type must be created to operate the myDatalogEASY IoT GW ATEX .*

2. Configure the created site/application according to your requirements (see "Site configuration" on page 84). If the site was not created based on an IoT application, you must determine the Data Descriptor and Device Logic via the "Control" configuration section (see "Control" on page 85).
3. Connect the antenna (see "Connecting the mobile network antennas" on page 74). The antenna is not included in the scope of delivery and must be ordered separately.
4. Establish a connection. If no script has been loaded in the device yet, this can be achieved by inserting the power supply unit as described in the chapter "Replacing the power supply unit" on page 266. If a Device Logic has already been loaded in the device, execute the operations provided in the Device Logic to trigger the establishment of a connection.

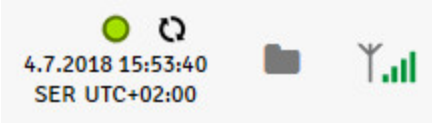
   Note: *If you are using a power supply unit without an integrated energy store, the external supply or charging voltage must be connected before inserting the power supply unit. Details on this are provided in the chapter "Connection of the power supply" on page 71.*



Inserting the power supply unit

1 Power supply unit	**2** Base unit

5. Wait until the measurement instrument list indicates that the device is connected to the server (rotating arrows).



   With the exception of the "Online" connection type (see "rM2M_TxSetMode()"), the time during which the myDatalogEASY IoT GW ATEX is connected to the server is very short. It can therefore also be checked whether the time stamp of the last connection (under the green status symbol) has been updated.

The following steps are only necessary, if you simultaneously want to test the measurement value acquisition and data transmission.

6. Complete all of the steps detailed in the chapter "Connection of the power supply" on page 71.

   Important note: *All wiring work must be performed in the de-energised state.*

7. You can use the "Reports" of the myDatanet server to check the data transmission (see "myDatanet Server Manual " 805002). The configuration of the Data Descriptor (see "Data Descriptor " on page 251) is required for this purpose.
8. Once you have completed the necessary preparations, initiate a transmission directly on the device if you have included this in your Device Logic. If you have not included an option to trigger a transmission, wait for the next scheduled data transmission.
9. Evaluate the incoming data.

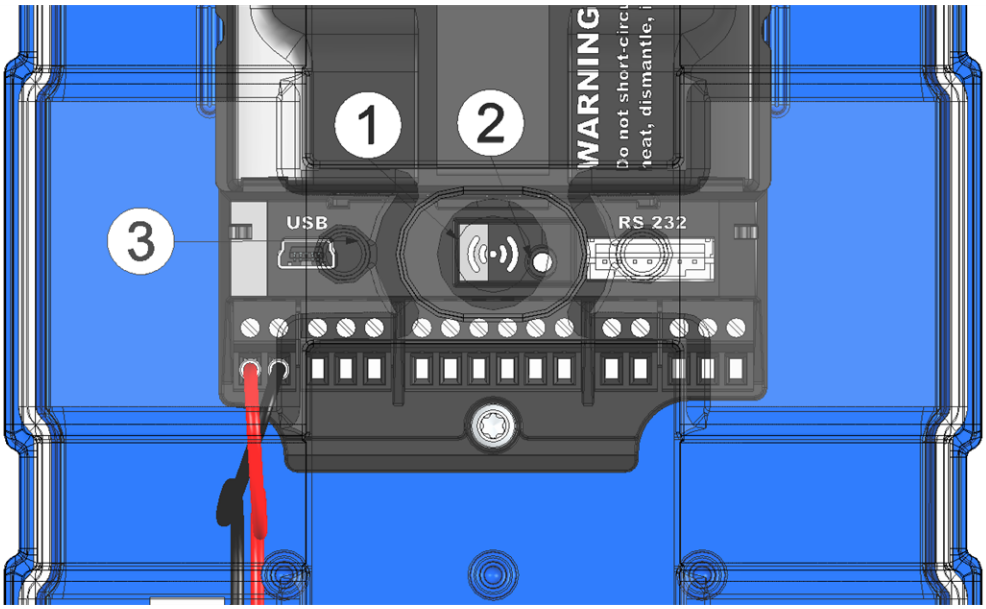
Chapter 10 User interfaces

The configuration of the myDatalogEASY IoT GW ATEX is carried out via the web interface on the myDatanet server (see "User interface on the myDatanet server" on page 84), which your responsible sales partner will provide to you.

10.1 User interface on the myDatalogEASY IoT GW ATEX

10.1.1 Operating elements

The operating elements of the myDatalogEASY IoT GW ATEX can still be operated when the housing is closed.



Operating elements

1 Solenoid switch	**2** Two-colour LED
3 Lid reed contact	

10.1.1.1 Solenoid switch

The solenoid switch is operated using the MDN Magnet (206.803) included in the scope of supply. The "Switch_Init()" function can be used to determine whether the solenoid switch is evaluated by the firmware or Device Logic.

Report	Operation by the user	Operation
Firmware	Press and hold for at least 3 sec. and then release	Initiation of the transmission
Device Logic	Press	Call up of the public function, for which the index is transferred to the "Switch_Init()"function
	Release	

10.1.1.2 Three-colour LED

The "Led_Init()" function can be used to determine whether the three-colour LED is controlled by the firmware or Device Logic. If the three-colour LED has been configured in such a way that it is controlled by the firmware, it is designed to signal the current operating state. The state of the three-colour LED can otherwise be controlled by the "Led_On()", "Led_Off()", "Led_Blink()" , "Led_Flash()" und "Led_Flicker()" Device Logic functions.

Operating states (Three-colour LED controlled by the firmware)

Three-colour LED	Colour	Description
Flickering	Green	Establishing connection
Lights up	Green	GPRS connection or USB connection established
Off	---	Normal operation/script processing until the next transmission

10.1.1.3 Lid reed contact

The lid reed contact is operated using the magnet contained in the housing cover. The function "LidCover_ Init()" can be used to activate the evaluation of the lid reed contact via the device logic.

Evaluation	Operation by the user	Operation
Device Logic	Press	Calling up of the public function whose index was passed to the "LidCover_Init" function
	Release	

10.2 User interface on the myDatanet server

10.2.1 Site configuration

> ***Note:*** *Depending on the respective user level, some of the configuration fields mentioned in the following sub-chapters may be hidden. In this case, please contact the administrator of the myDatanet server.*

Click on the name of the site in the list of sites to open the input screen for configuring the site (see "myDatanet Server Manual " 805002).

10.2.1.1 Site

Customer
Specifies to which customer the site is assigned

 **symbol**
Assign site to another customer

Name
Site designation (not relevant for the device or data assignment) [2-50 characters]

Device S/N
Serial number of the device that is linked to the site (device assignment!)

Application
Name of the IoT application based on which the site was created

Application version

Version number of the IoT application that is currently installed on the site. If the version number of the site is not the same as the version number of the device logic installed on the device then the version number of the device logic installed on the device is displayed in addition to the version number of the site.

Tags

List of tags that are already assigned to the site. This assignment can be cancelled by clicking on the cross next to the title of the tag. The input screen for assigning tags can be opened by clicking on the plus symbol.

10.2.1.2 Comments

Comments

Free comment field (is also displayed below the device type in the site/application list)

10.2.1.3 Control

Note: *This configuration section is not visible if this site was created based on an IoT application (see "myDatanet Server Manual " 805002).*

Device logic type	Off	Device logic deactivated	
	Pawn	Activates device logic processing	
Device logic source	Pawn source code	Device logic	Input window for editing the device logic that is loaded into the myDatalogEASY IoT GW ATEX (see "Device Logic" on page 131)
	Upload a compiled script	File upload	Selection of the device logic binary file (*.amx) that is uploaded to the myDatanet server and is loaded into the myDatalogEASY IoT GW ATEX during the next connection. The file path is only displayed as long as the input screen for configuring the site has not been closed.
Data descriptor	Input window for configuring the Data Descriptor (see "Data Descriptor " on page 251)		

10.2.1.4 Configuration 0 - Configuration 9

Note: *These configuration sections are only visible if the logical structure of the corresponding configuration data block was defined using the Data Descriptor (see "Data Descriptor " on page 251). The name of the configuration section is also defined via the Data Descriptor .*

These configuration sections ensure that the parameters from the customer's freely definable, independent memory blocks can be edited and displayed via the interface of the myDatanet server. For this purpose, the logical structure of the configuration data blocks must be defined with the help of the Data Descriptor (see "Data Descriptor " on page 251).

10.2.1.5 Alarm settings

Acknowledgement	Standard	The global server setting is used to determine whether alarms must be acknowledged automatically or manually.
	automatic	Alarms are acknowledged automatically as soon as all of the messages have been sent. If SMS that have a tariff with a delivery confirmation function have also been sent, acknowledgement is provided after delivery confirmation.
	manual	Alarms must be acknowledged by the user.
Transfer volume	Standard	The setting for the transfer volume alarm is taken from the global server settings.
	off	The transfer volume alarm is deactivated.
	individual	The level at which the transfer volume alarm should be triggered can be entered in the adjacent field in KiB.
Offline alarm after	alarm in the event that the device does not report for longer than the set time (00:00 alarm deactivated).	
Title user alarm 1	Freely selectable title for user-defined alarm 1. If the user-defined alarm 1 is triggered by a device connected to the site, the server will use this text to signal the alarm. The same applies to user-defined alarm 2 and 3.	
Title user alarm 2	Freely selectable title for user-defined alarm 2	
Title user alarm 3	Freely selectable title for user-defined alarm 3	

10.2.1.6 Basic settings

Time zone	Regional settings (not relevant for raw measurement data as this is stored in UTC)	
Daylight saving time	Configuration for automatic time adjustment	
	Standard	The configuration for the time adjustment is adopted by the global server setting.
	Off	Automatic time adjustment deactivated
	USA	Predefined setting for the American area
	EU	Predefined setting for the European area
Default report	Selection of the report that is loaded by clicking on the device link in the maps	
	Off	No report is loaded.
	"Name of a report"	The selected report is loaded.
Report template	Selection of the report template used to display the data when clicking on the symbol to display the measurement data, which is located in the list of sites/applications. Only the report templates in which the site/application type of the first wild card is compatible with the site/application that is currently being edited are displayed in the dropdown list. The symbol to display the measurement data is only displayed in the list of sites/applications if a report template has been selected.	
	(not assigned)	The symbol to display the measurement data is not displayed in the list of sites/applications.
	"Name of a report template"	Name of the report template used to display the measurement data
Change log configuration	Selection of which changes to the configurations should be logged	
	web api	Changes that were implemented via the server interface or REST-API are logged.
	web device api	Changes that were implemented via the server interface, by the device itself or the REST-API are logged.

10.2.1.7 FTP export settings

Note: *This configuration section is only visible if the "FTP Agent Extended" licence for the myDatanet server has been enabled.*

FTP export profile	off	FTP export deactivated
	"Name of an FTP export profile"	List with the FTP export profiles that were created on the myDatanet server (for creating an FTP export profile, see "myDatanet Server Manual " 805002).
Settings of the selected profile	Shows an overview of the most important parameters of the selected FTP export profile	
FTP directory	Makes overwriting the standard directory of the selected FTP export profile possible [0-100 characters]	
Last export	Time stamp of the last FTP export	

10.2.2 Device configuration

Note: *Several of the configuration fields in the following sub chapters may possibly be hidden depending on the respective user level. In this case, contact the myDatanet server administrator.*

You can reach the input screen for configuring the device by clicking on the serial number in the list of sites/applications (see "myDatanet Server Manual " 805002) or by clicking on the device name in the device name list (see "myDatanet Server Manual " 805002).

10.2.2.1 Comments

Comments

Free comment field (is also displayed below the site name in the site/application list)

10.2.2.2 Measurement instrument

Customer	Name of the customer to whom the measurement instrument is assigned
Tags	List of the tags that are already assigned to the measurement instrument. This assignment can be cancelled by clicking on the cross next to the title of the tag. The input screen for assigning the tags is opened by clicking on the plus symbol. This enables existing tags to be assigned and new tags to be created.
Serial number	Serial number of the instrument
Instrument class	The instrument class of the site and instrument must match for an instrument to be able to be connected to a site. Once the instrument has been created via the server interface, the instrument class can only be changed up until the first connection of the instrument to the server. If an instrument class, that does not match the actual class of the instrument, is selected when the instrument is created it is automatically corrected during the first connection.
Telephone number	Telephone number of the SIM card. The control SMS messages (e.g. wakeup) are sent to this number. Format: +43555837465
Instrument flags	Additional information regarding the instrument class (for internal use)
Firmware version	Current software version installed on the measurement controller
Last connection	In each case, the last time stamp of the affected operation
Last wakeup	
Last disconnection	
Last transmission error	
Last Aloha connection	
Wakeup SMS count	Number of wakeup SMS sent to this device since the last connection. This counter is reset at/during each successfully established connection.

Device Logic sync	Productive	If the Device Logic installed on the device and saved on the server do not match, the Device Logic saved on the server is loaded in to the device.
	Development (sync)	The Device Logic on the device and server are synchronised. The one with the latest time stamp is transferred to the other one.
	Development (no sync)	The Device Logic on the device and server are not synchronised.
Firmware update	Off	Firmware update is deactivated.
	On	As soon as a new version of the selected firmware type is available, this is installed immediately.
	Even if tag is missing	Firmware is also transferred to the device if the device has not transmitted the current firmware version to the server (NOT RECOMMENDED!).
	Allow downgrade	Facilitates the installation of an older firmware version than the one on the device (NOT RECOMMENDED!)
	Once	Performs a single firmware update. If no new firmware is available or the firmware was installed successfully, the firmware update is automatically switched to "OFF".
	Ignore	The firmware update is deactivated and no information is provided about available firmware updates.
Firmware type	Released	Only firmware versions that have successfully undergone internal and field testing are installed (this practically eliminates malfunctions).
	Release candidate	Only firmware versions that have successfully undergone internal testing are installed (malfunctions cannot be excluded).
	Beta release	Even firmware versions that have not successfully undergone all of the internal tests are installed (malfunctions may occur).
Identification	String specifying the hardware platform implemented in the device and the corresponding hardware version (i.e. the rapidM2M module identification).	
Hardware version	Hardware version of the myDatalogEASY IoT GW ATEX	

10.2.2.3 GPRS

SIM tariff
Selected SIM tariff

Chapter 11 DeviceConfig

11.1 General

The DeviceConfig configuration program can be downloaded free of charge from the following website:

www.microtronics.com/deviceconfig

The tool is used for configuration, maintenance, fault analysis and synchronisation purposes. It is compatible with all myDatanet devices equipped with a USB interface, wireless M-bus interface or Bluetooth Low Energy.

The requirements regarding configuration and maintenance vary depending on the type of device. To ensure simple and intuitive operation, the user interface of the DeviceConfig therefore automatically adjusts to the relevant device that is connected. In addition to the standard functions, the tool also supports device-specific processes (e.g. calibration, zero point adjustment).
The DeviceConfig enables you to complete the following tasks:

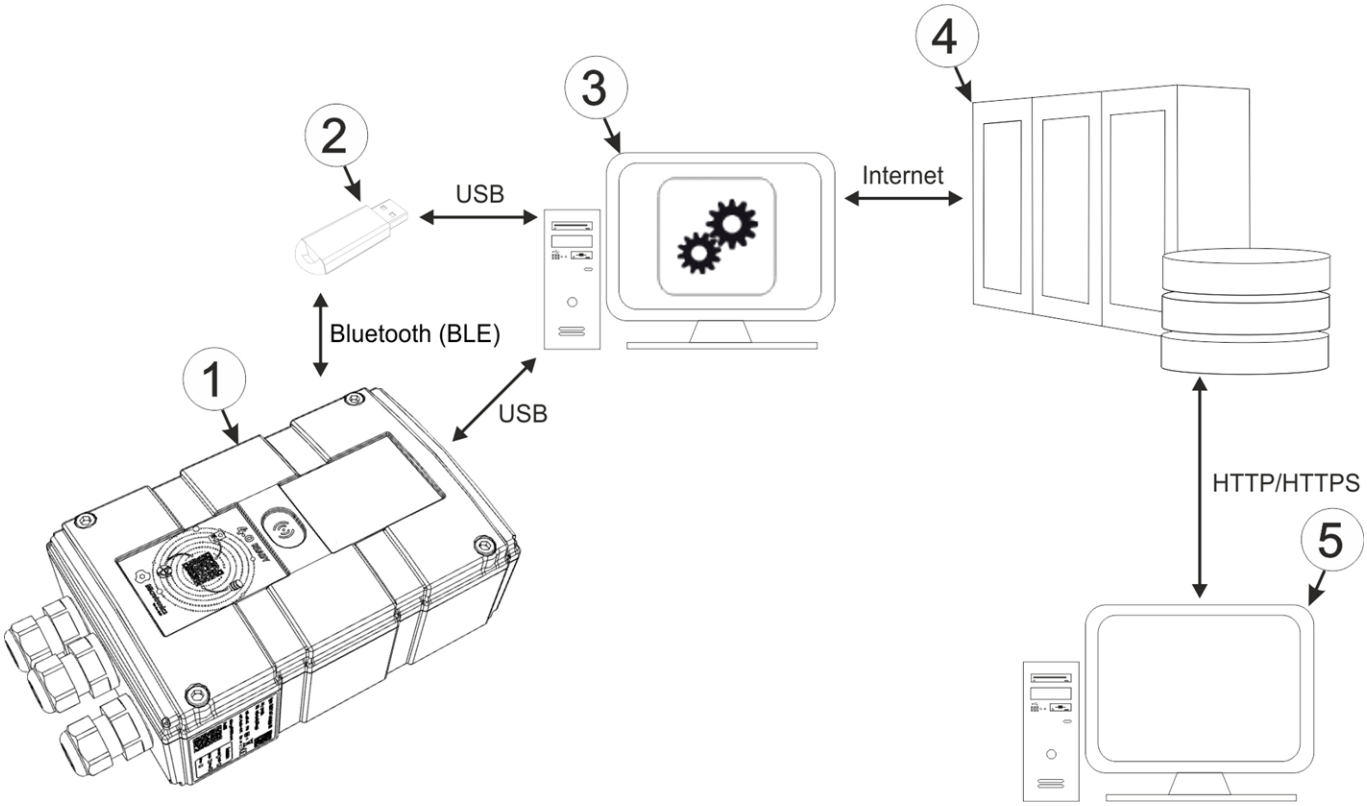
- Synchronisation of measurement data and configuration between device and server (specifically for devices without GSM/GPRS modem)
- Basic configuration of the device (e.g. measurement and transmission cycle)
- Read out and analysis of the device log
- Calibration, trimming and zero point adjustment (special knowledge and password required)
- Update the firmware

11.2 Prerequisites

Interfaces	1 x USB
Operating system	Win XP Windows Vista Windows 7 Windows 8 Windwos 10
Internet connection	Recommended
Required disk space	approx. 50 MB

11.3 Functional principle

The following description specifically refers to the use of the configuration program DeviceConfig in conjunction with the myDatalogEASY IoT GW ATEX .



Functional principle

1	myDatalogEASY IoT GW ATEX	**4**	myDatanet server
2	USB BLE-Adapter	**5**	Client that accesses the interface of the myDatanet server via the web browser
3	PC with the DeviceConfig configuration program installed		

Important note: *The USB interface is intended as a mainenance interface that is only accessible once the housing has been opened. The manufacturer is not liable for damage to the device, e.g. due to the ingress of moisture, which can be attributed to opening the housing.*

The DeviceConfig configuration program either communicates wirelessly (Bluetooth Low Energy) with the myDatalogEASY IoT GW ATEX using the USB BLE-Adapter (300685) or directly with the myDatalogEASY IoT GW ATEX via a USB connection.

The functions provided with the DeviceConfig configuration program include:

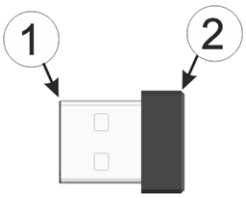
- Synchronisation of measurement data and configuration between device and server (see ""Sync" tab" on page 104)
- Switching between the integrated SIM chip and the external SIM card (see ""GSM" tab" on page 99)
- Read out and analysis of the device log (see ""Log" tab" on page 101)

- Update the firmware(see ""Firmware" tab" on page 103)
- Entering the activation code in order to unlock chargeable features (see ""Features" tab" on page 104)

As soon as the data has been transferred to the myDatanet server, it is available via all of the server's interfaces (e.g. HTTP/HTTPS, as illustrated in the functional principle above) in the same way as the data from all of the other myDatanet devices.

11.3.1 USB BLE-Adapter

The USB BLE-Adapter (300685) is not included in the scope of delivery of the myDatalogEASY IoT GW ATEX . It is required because standard PCs and laptops often do not have a Bluetooth Low Energy interface that is necessary for communication with the myDatalogEASY IoT GW ATEX .



USB BLE-Adapter

1 USB connector (type A)	**2** Antenna

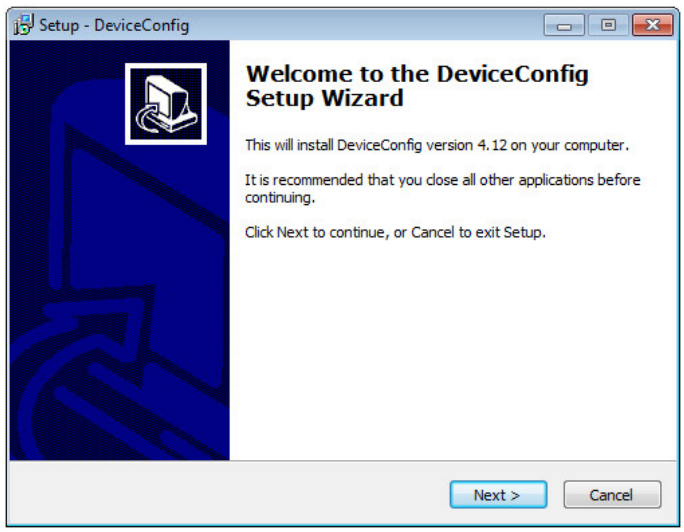
The use of USB extension cables of up to 180 cm is possible without any problems.

11.4 Installation

The following chapter describes the installation process in Windows 7.

1. Execute the *"InstDeviceConfig.exe"* file to start the installation process.

   Note: *Only connect the device or USB BLE-Adapter (300685) to your PC once the installation process has completed as the required drivers are only installed during this process.*



DeviceConfig setup wizard

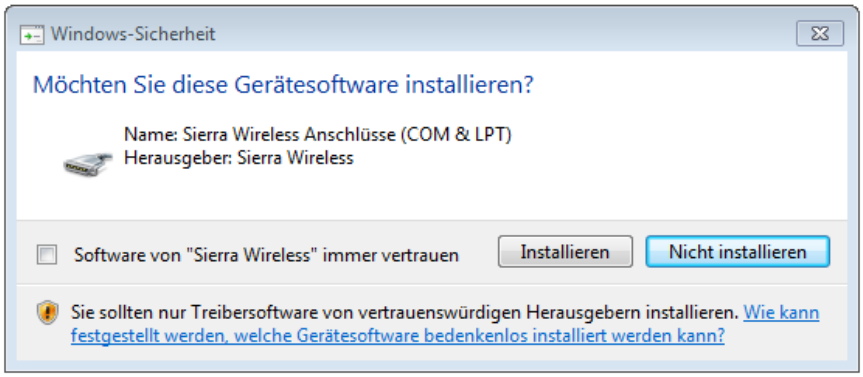
2. Follow the instructions of the setup wizard until the following screen is displayed. The following drivers must be installed to ensure correct operation.



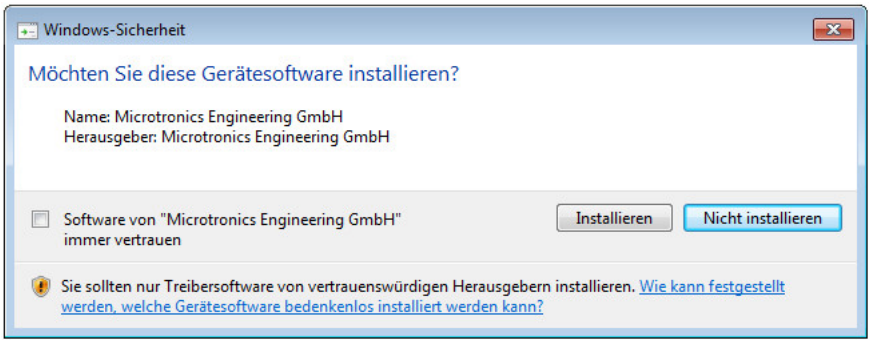
Installation of the USB drivers for the devices



Installation of the drivers for the USB BLE-Adapter

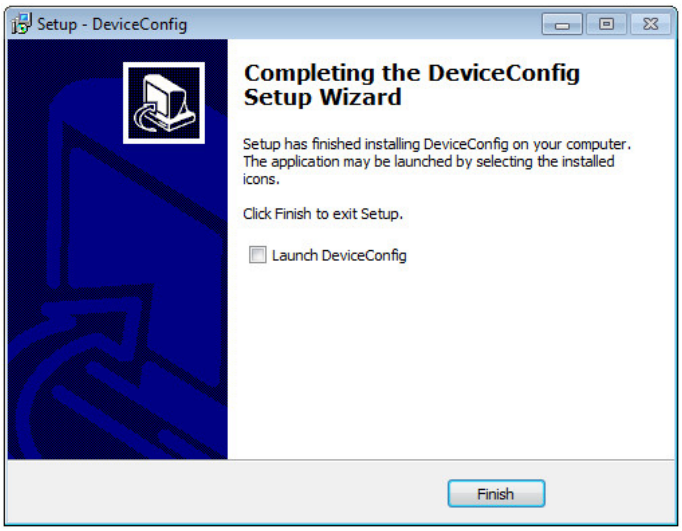


Installation of the USB drivers for the devices on a M1 basis



Installation of the USB drivers for the devices on a M2/M3 basis

3. Once the following screen is displayed, close the installation process by clicking on the *"Finish"* button.



Complete the setup

11.4.1 Installing USB BLE-Adapter driver

> ***Note:*** *Information on the USB BLE-Adapter (300685) is provided in chapter "USB BLE-Adapter " on page 93.*

The following chapter describes the installation process in Windows 7.

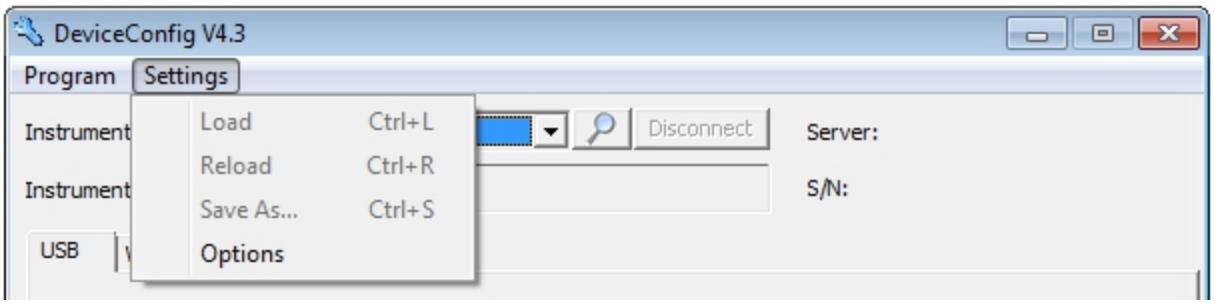
1. Complete all of the steps described in chapter "Installation" on page 93.
2. Close the DeviceConfig configuration program, if, when completing the installation, you selected the option for the program to be started following completion of the installation process.
3. Connect the USB BLE-Adapter (300685) to a free USB port on your PC. From Windows Vista onwards, the driver will be installed automatically. Instructions on installing the driver on older versions of Windows are included in the user manual for the DeviceConfig ("myDatanetDeviceConfig Manual " 805004).

   > ***Note:*** *If possible, always use the same USB port, as the driver will have to be installed for every USB port used to connect the USB BLE-Adapter (300685) to the PC for the first time.*

4. Wait until the driver installation process is complete. This can take several minutes depending on the performance of your PC.

11.5 Menu of the DeviceConfig

11.5.1 Settings

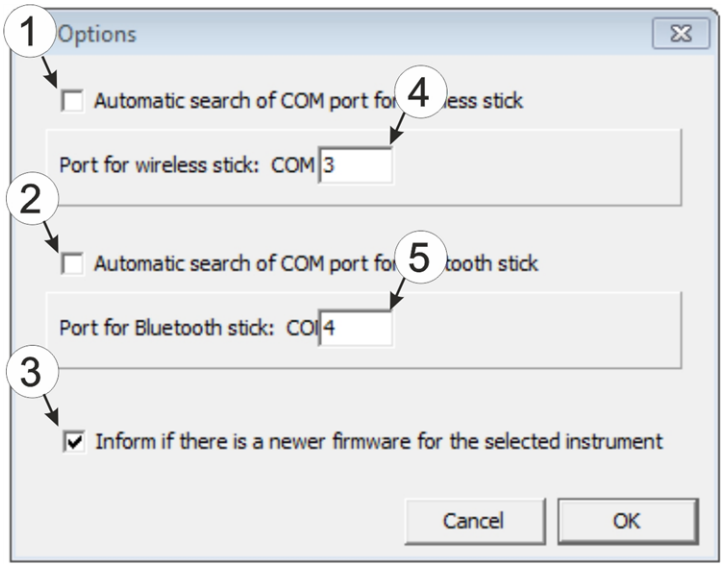


"Settings" menu item

11.5.1.1 Options

The settings for the COM ports to which the USB radio transmitter (206.657) or the USB BLE-Adapter (300685) are connected can be specified and the automatic search for the available firmware versions can be activated or deactivated via the "Settings -> Options" menu item.

The USB radio transmitter (206.657) is required for myDatanet devices that are connected to the PC via a wireless M-bus, while the USB BLE-Adapter (300685) is required for devices that are connected to the PC via Bluetooth Low Energy. Information on whether your device supports one of these connection methods is provided in the user manual for the respective device.

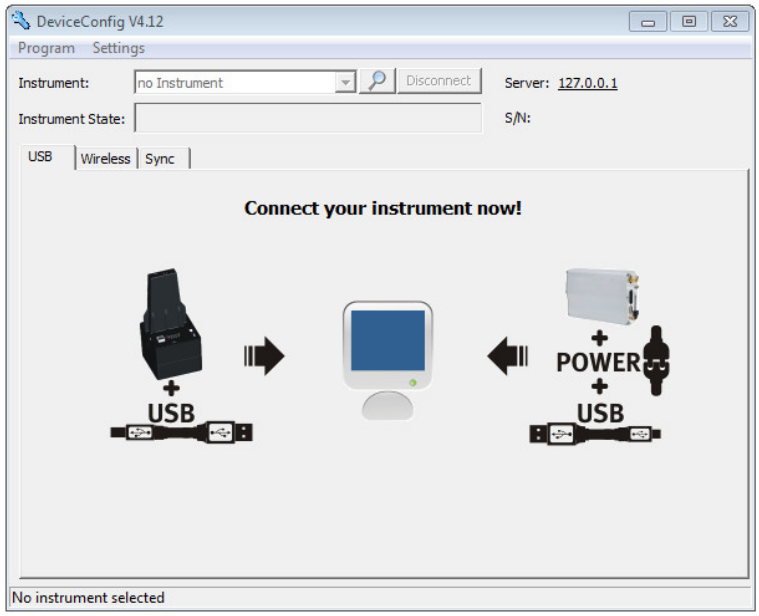


"Settings -> Options" menu item

1	Activates/deactivates the automatic search for the USB radio transmitter (206.657) on all of the available COM ports	**4**	COM port that is connected with the USB radio transmitter (206.657) (only visible when the automatic search is deactivated)
2	Activates/deactivates the automatic search for the USB BLE-Adapter (300685) on all of the available COM ports	**5**	COM port that is connected with the USB BLE-Adapter (300685) (only visible when the automatic search is deactivated)
3	Activates/deactivates the automatic search for available firmware versions		

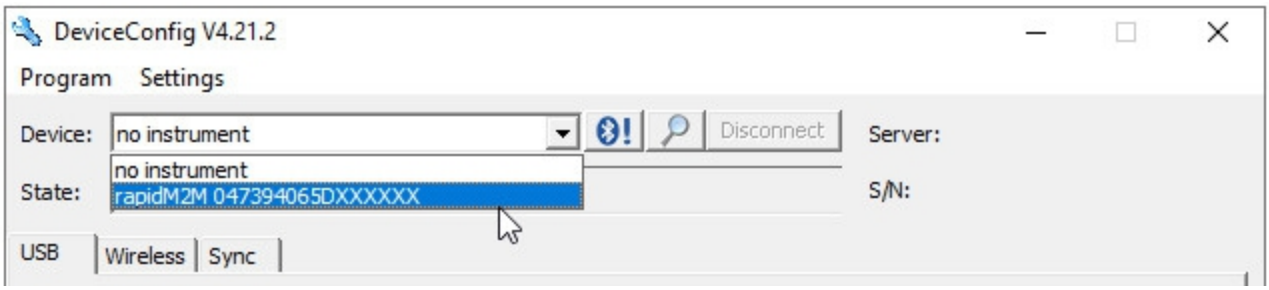
11.6 Connecting a Device via USB

1. Start the DeviceConfig configuration program.



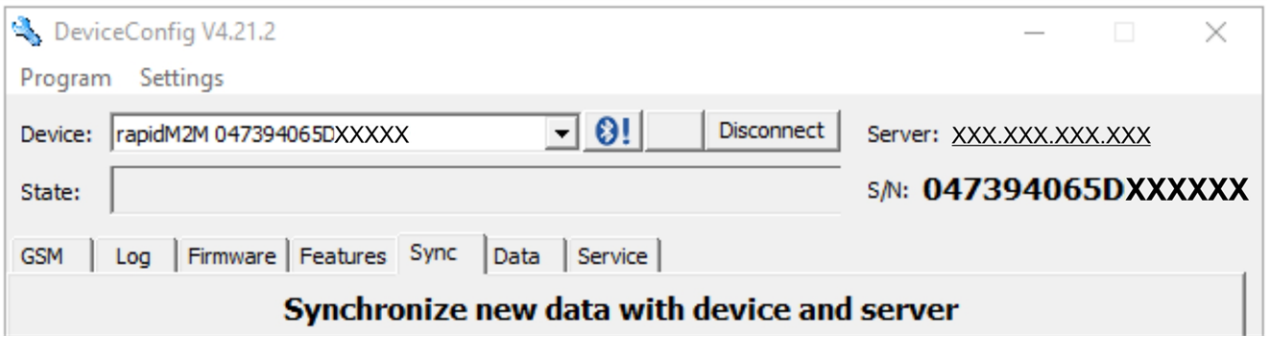
DeviceConfig

2. Connect the myDatalogEASY IoT GW ATEX to the PC using a USB cable.
3. Select your device based on the serial number from the list of devices found.



List of devices found

4. Wait until the DeviceConfig has received the configuration of the device. Depending on the device, additional tabs may be displayed.

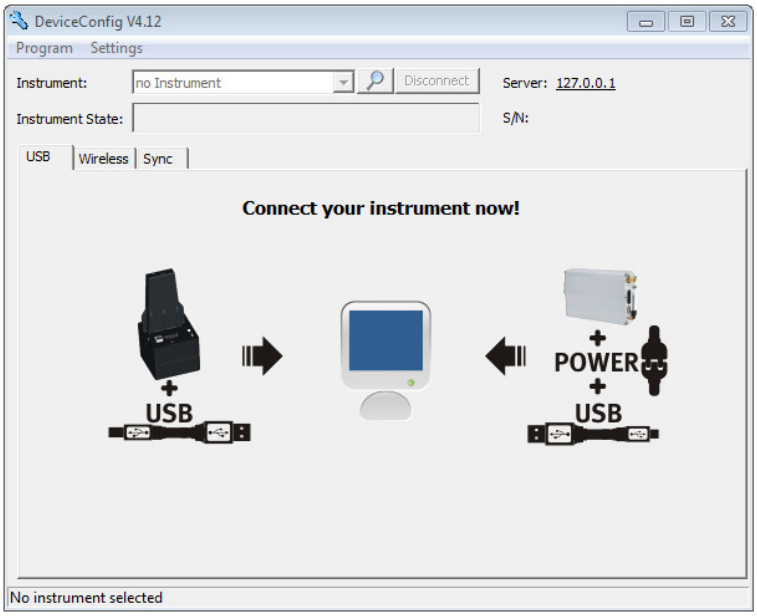


"Sync" tab when actively connected to the myDatalogEASY IoT GW ATEX

11.7 Connecting a Device via Bluetooth Low Energy

The USB BLE-Adapter (300685) is required to establish a connection to a device with a Bluetooth Low Energy interface. First of all complete the steps described in chapter "Installing USB BLE-Adapter driver" on page 95 to install the drivers required to operate the USB BLE-Adapter .

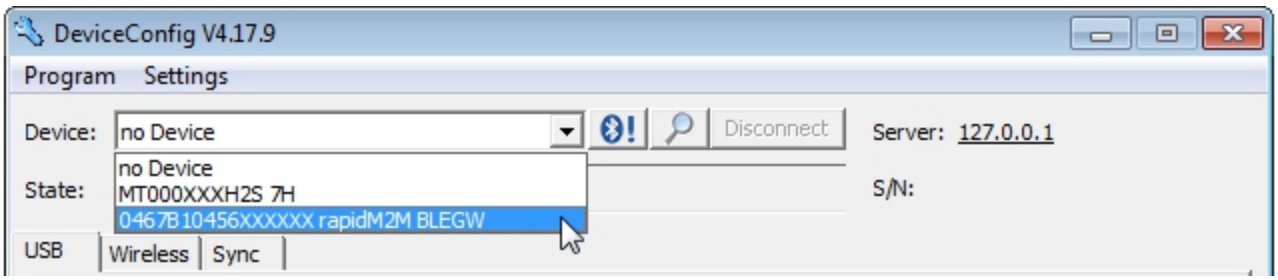
1. Connect the USB radio transmitter (206.657) to the USB interface of your PC.
2. Start the DeviceConfig configuration program.



DeviceConfig

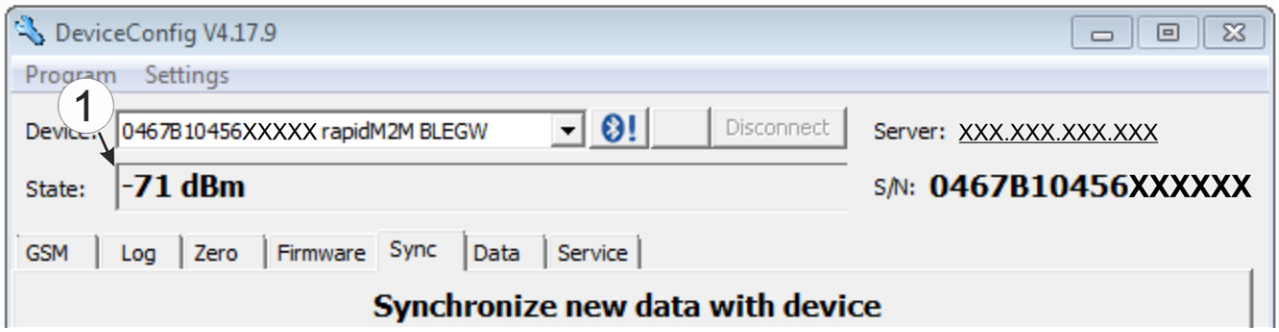
3. Select your device based on the serial number from the list of devices found.

> ***Important note:*** *Please note that, depending on the environmental conditions, the range of the radio transmitter of the myDatalogEASY IoT GW ATEX is max. 20m .*



List of devices found

4. Wait until the DeviceConfig has received the configuration of the device. Depending on the device, additional tabs may be displayed.



"Sync" tab when actively connected to the myDatalogEASY IoT GW ATEX

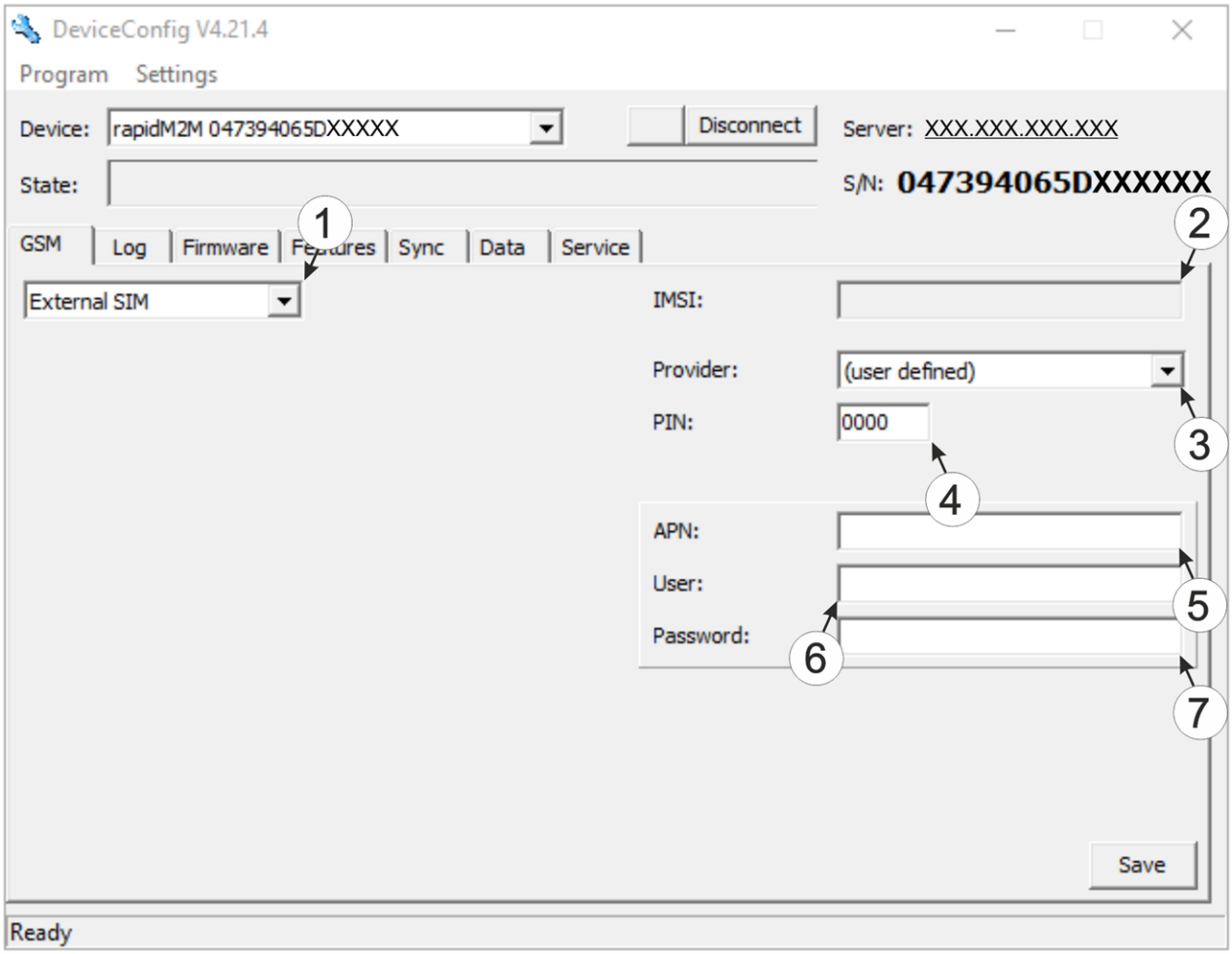
1 Wireless signal level [dBm]	

Note: *To ensure a stable connection, the wireless signal level should be higher than -90dBm , i.e. for example -85dBm . This is achieved by reducing the distance between the myDatalogEASY IoT GW ATEX and USB BLE-Adapter , and avoiding obstacles such as walls and similar.*

11.8 "GSM" tab

This tab provides the option of switching between the integrated SIM chip and an external SIM card inserted in the SIM slot. If the external SIM card has been selected then the APN settings (APN, username and password) and the PIN code (if required by the SIM card) can be entered via this tab and transferred to the myDatalogEASY IoT GW ATEX . Here it is possible to either enter the APN settings manually or to select one of the providers from the drop-down list and thus to use the settings stored for the provider in the DeviceConfig .

Note: *The manufacturer assumes no liability for the correctness of the APN settings deposited in the DeviceConfig (APN, username and passwort). In case of doubt please contact the provider of your external SIM card and enter the APN settings (APN, username and passwort) manually via the corresponding fields.*

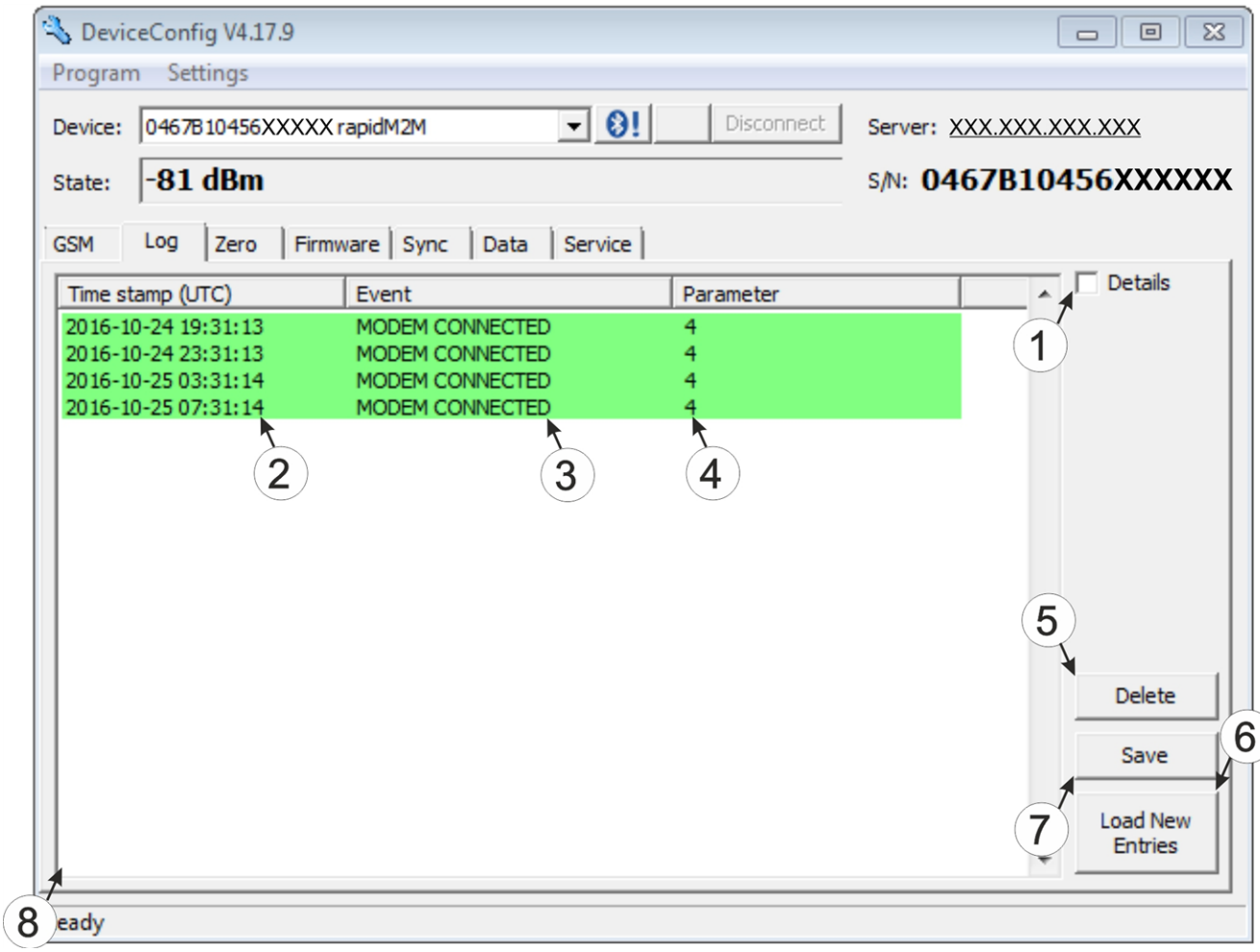


"GSM" tab

1	Drop-down list for selecting whether the internal SIM chip or the external SIM card should be used ***Note:*** *If "Internal SIM" has been selected then all other selection fields are hidden.*
2	IMSI of the external SIM card inserted in the SIM slot
3	Drop-down list for selecting the provider from which the external SIM card has been delivered ***Note:*** *The input fields for "APN", "user" and "password" for manual input of the APN settings are only displayed if the "(user defined)" entry has been selected in this drop-down list.*
4	PIN code
5	Access point (APN) that should be used for the connection
6	User name for dial-up via the access point
7	Password for dial-up via the access point

11.9 "Log" tab

This tab is designed to manage log entries. It enables the entries to be loaded from the myDatalogEASY IoT GW ATEX , to be saved as a *.tsv file and entries to be deleted from the memory of the myDatalogEASY IoT GW ATEX .

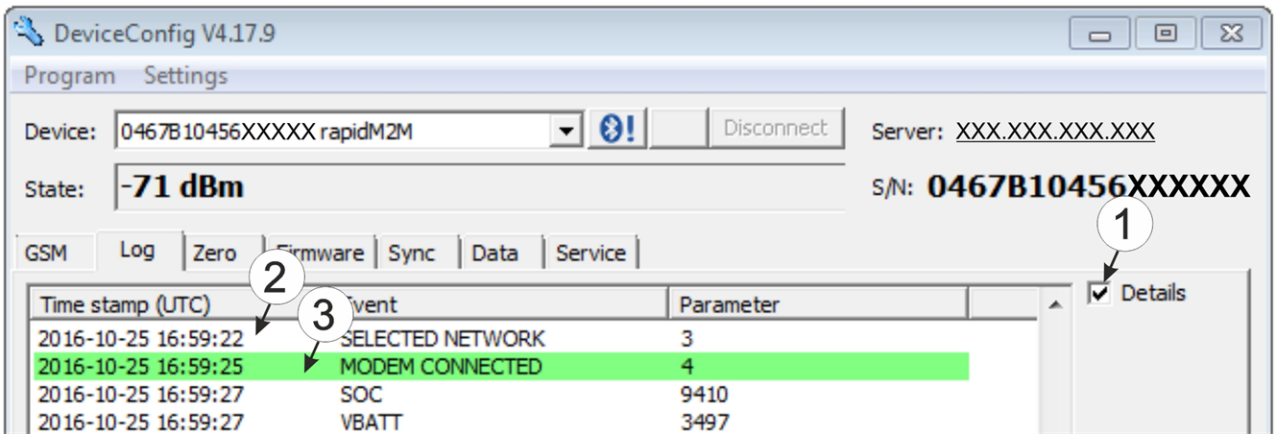


"Log" tab

1	Activates the detailed display of the log entries	**5**	Deletes the log entries from the memory of the device
2	Time stamp of the log entry	**6**	Loads the log entries from the device
3	Log entry	**7**	Saves the loaded log entries as a tsv file
4	Parameter of the log entry	**8**	Window to display the loaded log entries

The coloured highlighting indicates how crucial the log entry is. The white, informative log entries are only displayed when the detailed display of the log entries is activated (see ""Log" tab with detailed view activated" on page 102).

Colour	Evaluation
White	Information regarding the current operating state
Green	
Light blue	
Blue	
Purple	
Grey	
Yellow	Uncritical error
Red	Critical error



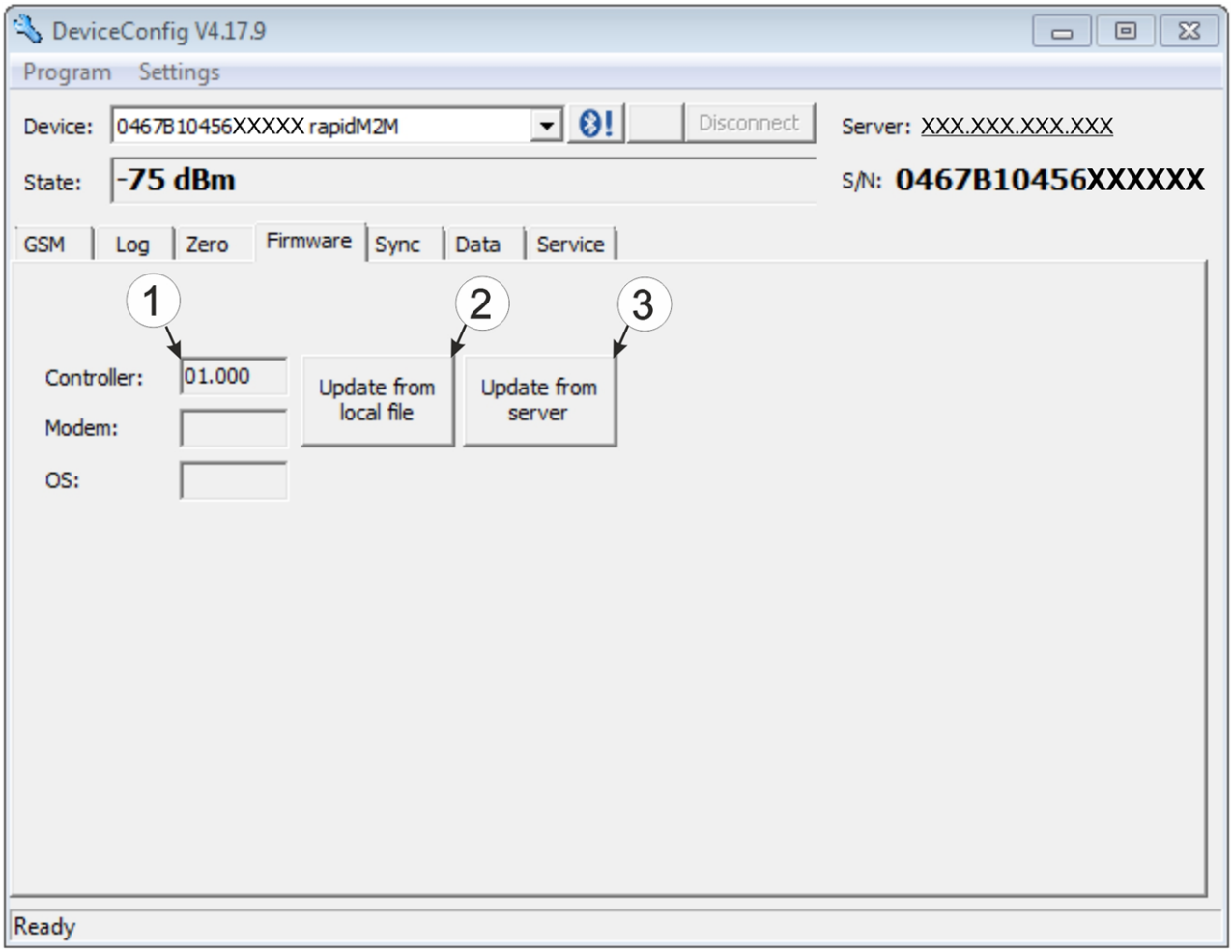
"Log" tab with detailed view activated

1 Activates the detailed display of the log entries	**3** Log entry that is always displayed
2 Informative log entry that is only visible if the detailed display is activated	

11.10 "Firmware" tab

This tab enables firmware to be installed directly via the USB interface or the Bluetooth Low Energy interface. There are two available ways to update the firmware:

- Using a previously downloaded firmware package
- By directly loading from the myDatanet server

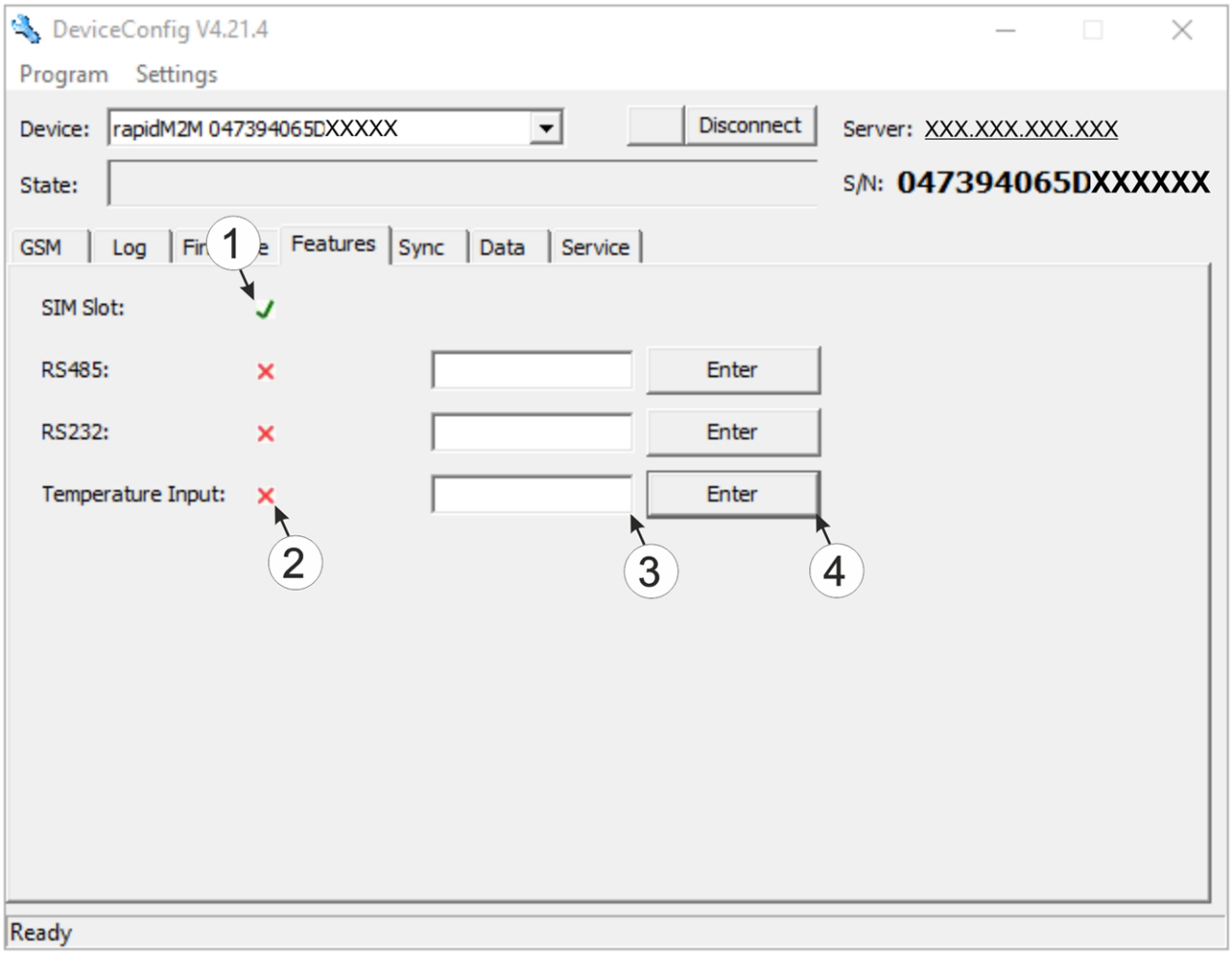


"Firmware" tab

1	Currently installed software version	**3**	The firmware is loaded directly from the server and installed on the device.
2	Button to install a previously downloaded firmware package		

11.11 "Features" tab

This tab provides the option of unlocking chargeable features by entering the activation code. It also offers an overview of the additional features that can be activated and which of these have already been activated.



"Features" tab

1	Feature has already been unlocked	**3**	Input field for the activation code
2	Feature is not unlocked.	**4**	Button for confirming the activation code and unlocking the feature

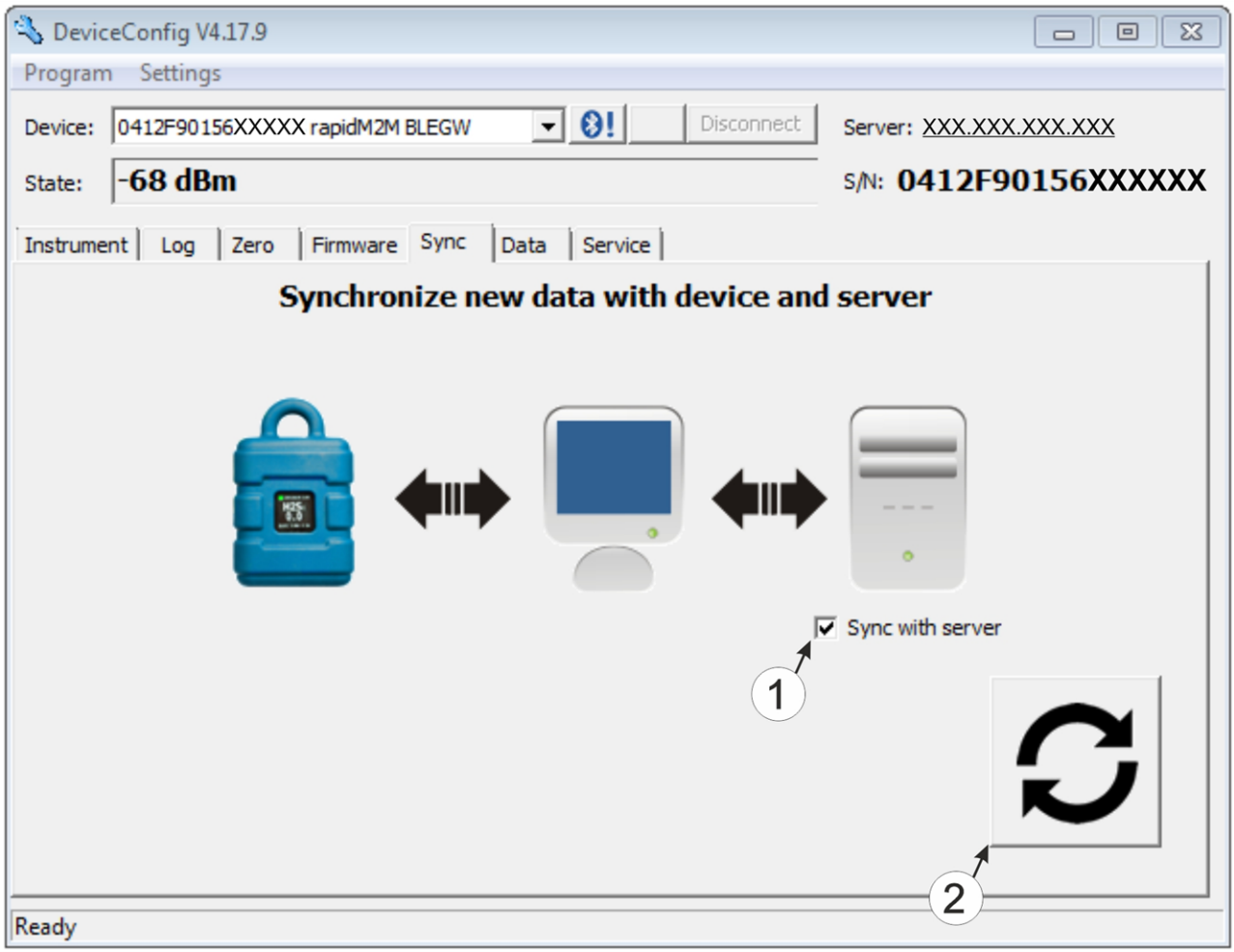
11.12 "Sync" tab

This area is designed to synchronise measurement data and configurations between myDatalogEASY IoT GW ATEX , DeviceConfig and myDatanet server. The "Sync" tab is also available if there is no connection (USB, wireless M-bus or Bluetooth) to a device.

Detailed instructions on completing the synchronisation is provided in chapter "Synchronisation with the myDatanet server" on page 107.

11.12.1 Existing connection to the myDatalogEASY IoT GW ATEX

If there is an existing connection to the myDatalogEASY IoT GW ATEX , there is an option to only synchronise the measurement data and configurations with the DeviceConfig configuration program for local processing or to transfer them to the myDatanet server. In the event that your PC is not connected to the Internet when reading out the data, you can initially synchronise the measurement data and configurations of the myDatalogEASY IoT GW ATEX with the DeviceConfig configuration program. As soon as your PC establishes a connection to the Internet, for example when you return to the office, you can then complete the synchronisations between the DeviceConfig and myDatanet server (see "No connection to a device" on page 106).

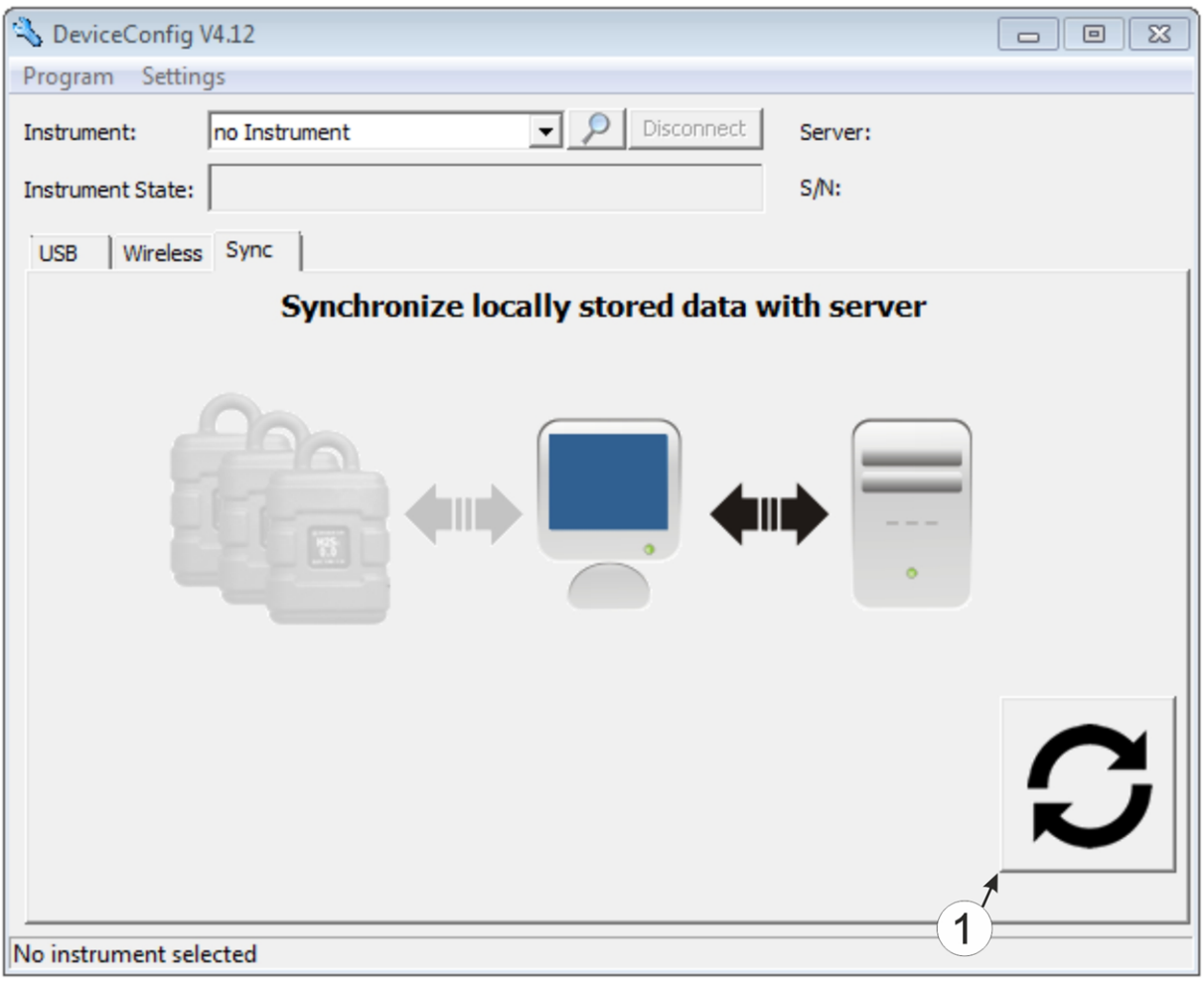


"Sync" tab when connected to the myDatalogEASY IoT GW ATEX

1	Checkbox to determine whether the measurement data and configurations should also be synchronised with the server when clicking on the Sync button. ***Note:*** *This checkbox is only displayed if your PC is already connected to the Internet.*
2	Button to trigger synchronisation

11.12.2 No connection to a device

This option can be used to complete the synchronisation retrospectively, if no connection to the Internet was possible while reading out the measurement data and configurations from the myDatalogEASY IoT GW ATEX .



"Sync" tab without connection to a device

1	Button to trigger synchronisation During this process, the measurement data and configurations for all of the devices that the DeviceConfig configuration program has saved locally are synchronised with the myDatanet server.

11.13 Recommended procedure

11.13.1 Synchronisation with the myDatanet server

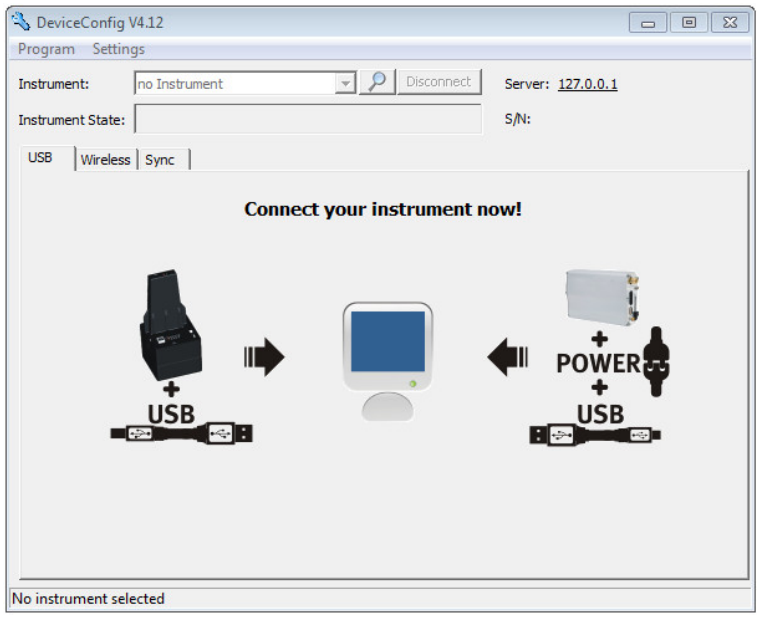
To ensure a more comprehensive management and display of the data, the DeviceConfig configuration program also provides the option of forwarding the measurement data and configurations to a central myDatanet server. The next two chapters describe the possible scenarios when reading out the data from the myDatalogEASY IoT GW ATEX .

More information on the functions of the server is provided in the server manual ("myDatanet Server Manual " 805002).

11.13.1.1 Internet connection available when reading out the data

The following process describes how you can not only synchronise the data with the DeviceConfig configuration program but also with the myDatanet server. A site must already be assigned to the myDatalogEASY IoT GW ATEX on the myDatanet server for this purpose. Detailed instructions are provided in chapter "Creating the site" on page 121. Another prerequisite for this is that your PC is connected to the Internet while reading out the data from the myDatalogEASY IoT GW ATEX . If this is not possible, follow the procedure described in chapter "No Internet connection when reading out the data" on page 111.

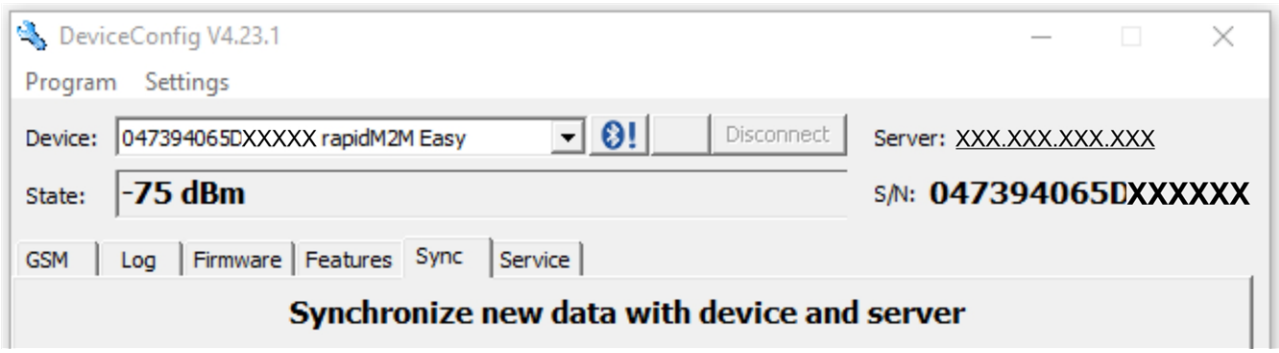
1. Connect the USB BLE-Adapter (300685) to the USB interface of your PC.
2. Start the DeviceConfig configuration program.



DeviceConfig

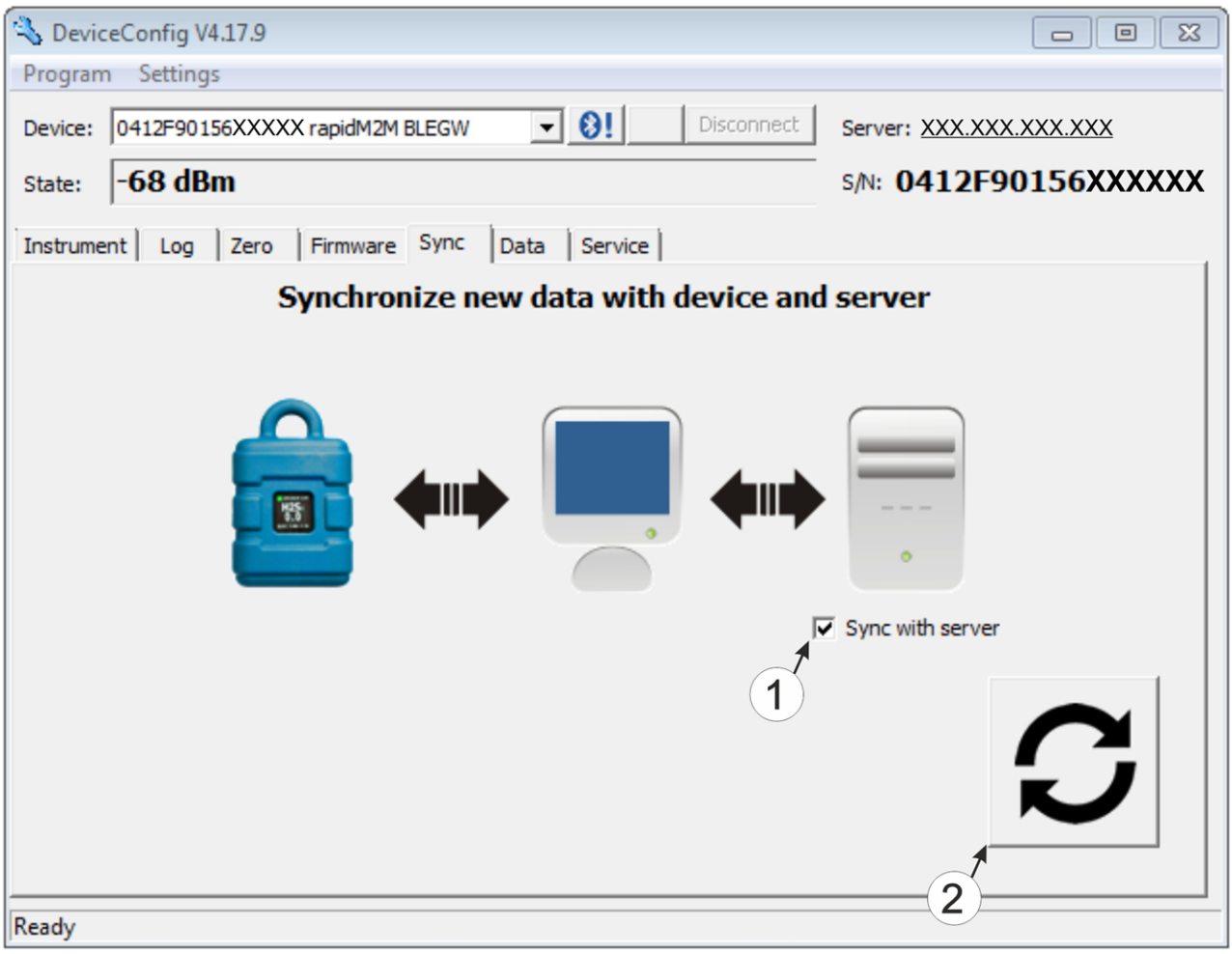
3. Connect the myDatalogEASY IoT GW ATEX to the PC using the USB BLE-Adapter (300685) supplied (see "Connecting a Device via Bluetooth Low Energy" on page 98).

4. Additional tabs are displayed if the connection was established successfully. Now select the "Sync" tab.



myDatalogEASY IoT GW ATEX specific tab

5. Place a tick in the "Sync with server" checkbox. This checkbox is only visible if your PC is currently connected to the Internet.

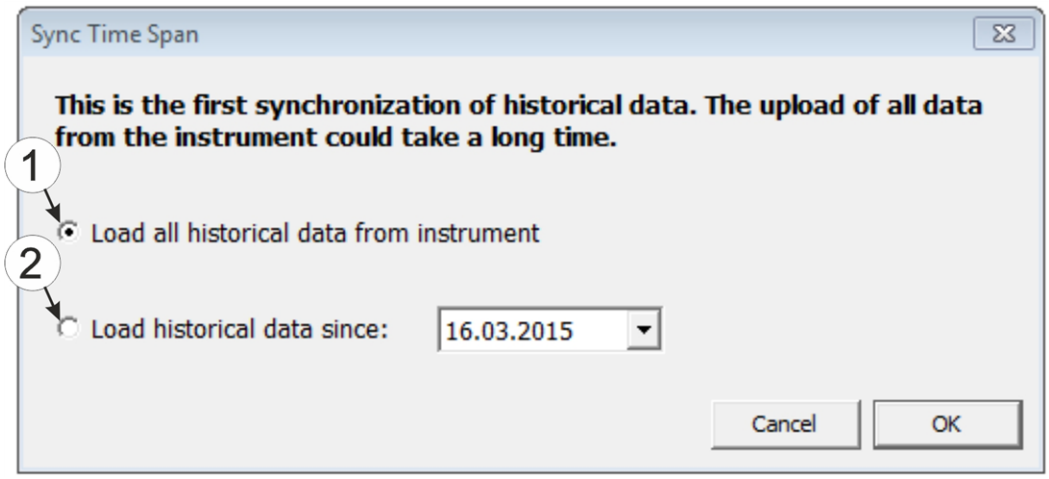


"Sync" tab when connected to the myDatalogEASY IoT GW ATEX

1	Checkbox to determine whether the measurement data and configurations should also be synchronised with the server when clicking on the Sync button. ***Note:*** *This checkbox is only displayed if your PC is already connected to the Internet.*
2	Button to trigger synchronisation

6. Click on the button to trigger the synchronisation (see ""Sync" tab when connected to the myDatalogEASY IoT GW ATEX " on page 108).

   When you read out the data for the first time from a myDatalogEASY IoT GW ATEX , you can choose whether all of the saved data or only the data from a certain date onwards are read out from the myDatalogEASY IoT GW ATEX . During the following synchronisation processes, the DeviceConfig configuration program always reads out the data from the last synchronised measurement data record onwards.

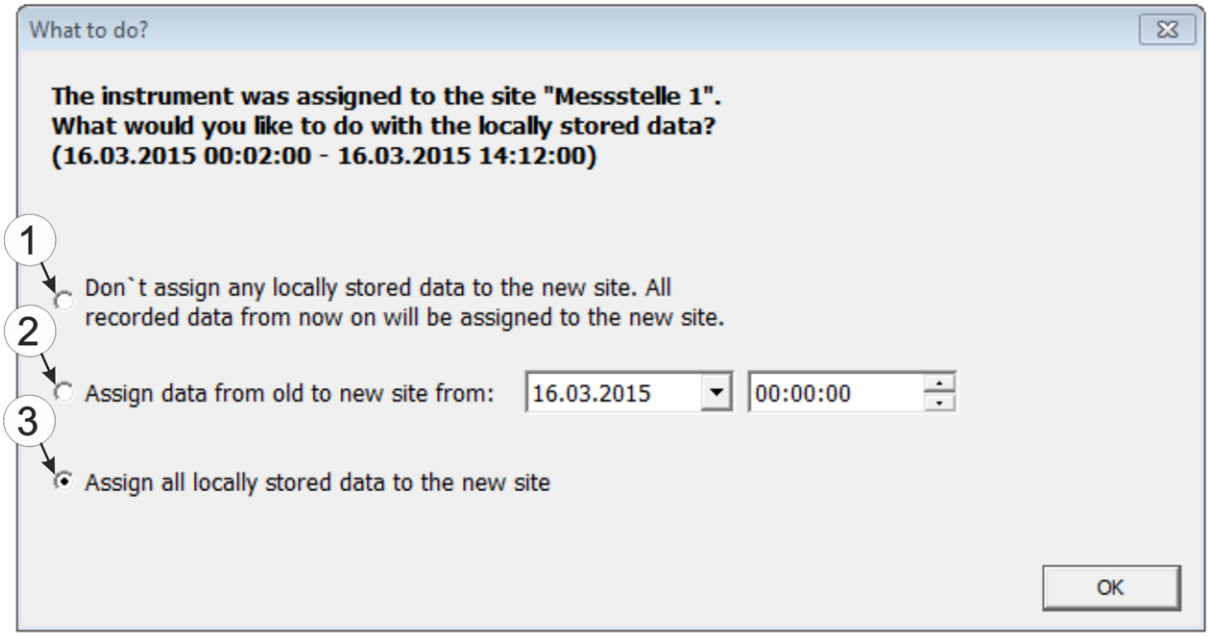


Selecting the time period for which the data should be read out (only during first synchronisation)

1	Read out all of the saved data ***Note:*** *Reading out all of the saved data can take several hours depending on the number of saved measurement data records.*
2	Only read out the data from the selected date onwards. The data is always read out from 00:00 am of the selected day. ***Important note:*** *Following completion of the synchronisation it is no longer possible to read out data before the selected date.*

If the DeviceConfig configuration program determines that the myDatalogEASY IoT GW ATEX has been assigned to a new or different site on the myDatanet server, you can decide what you would like to do with the data that is already saved locally. The following screenshot provides an overview of the available options that can be selected.

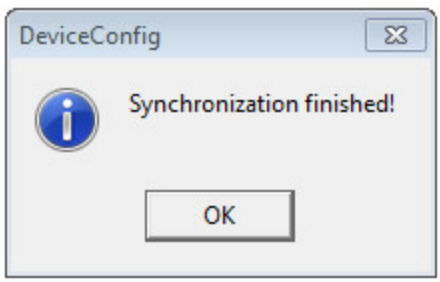
Important note: *If a site already contains data, then only the measurement data that is newer than the current measurement data record for the site is synchronised.*



Selecting how to proceed with the locally saved data (only if the device has been assigned a new or different site)

1	The locally saved data is not assigned to the new site. Only the measurement data that is read out from the current point in time is assigned to the new site. After clicking on "OK", the option to select from which time period onwards the data should be read out opens. The measurement data is read out from the myDatalogEASY IoT GW ATEX in accordance with the selection and assigned to the new site.
2	The locally saved data is assigned to the new site from the selected point in time onwards.
3	All locally saved data is assigned to the new site.

7. Wait until the DeviceConfig configuration program indicates that the synchronisation process is complete.



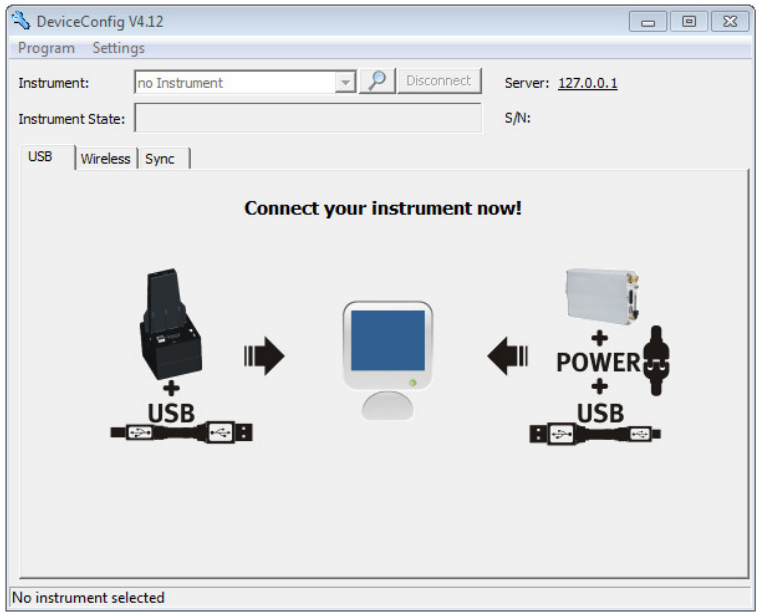
Synchronisation completed

11.13.1.2 No Internet connection when reading out the data

> ***Important note:*** *The method described in the following requires that the myDatalogEASY IoT GW ATEX myDatanet has already been assigned a site on the myDatanet server. Detailed instructions are provided in chapter "Creating the site" on page 121.Additionally, a synchronisation must already have been completed during which the DeviceConfig configuration program simultaneously established a connection to the myDatalogEASY IoT GW ATEX and the myDatanet server (see "Internet connection available when reading out the data" on page 107).*

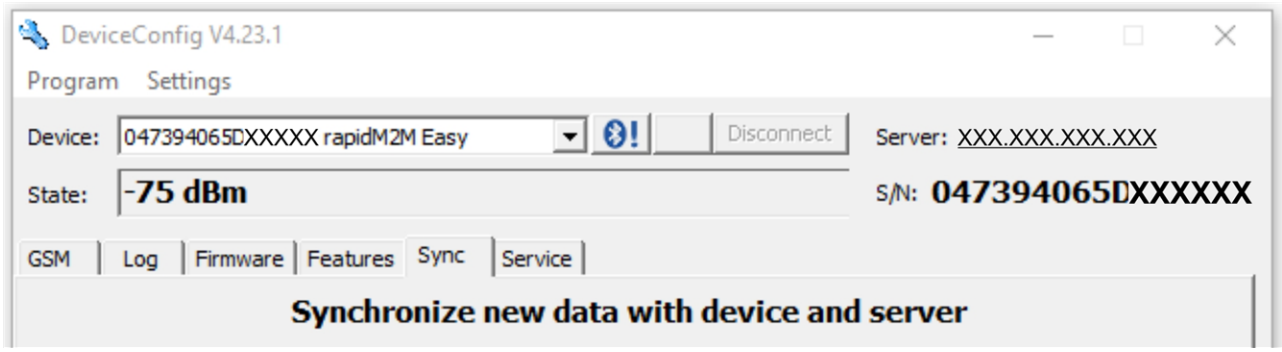
This procedure is recommended, if an Internet connection cannot be established on the site while reading out the data from the myDatalogEASY IoT GW ATEX . During this procedure, the data on the site is initially only synchronised with the DeviceConfig configuration program. The data is then transferred to the myDatanet server at a later date when your PC has re-established a connection to the Internet.

1. Connect the USB BLE-Adapter (300685) to the USB interface of your PC.
2. Start the DeviceConfig configuration program.



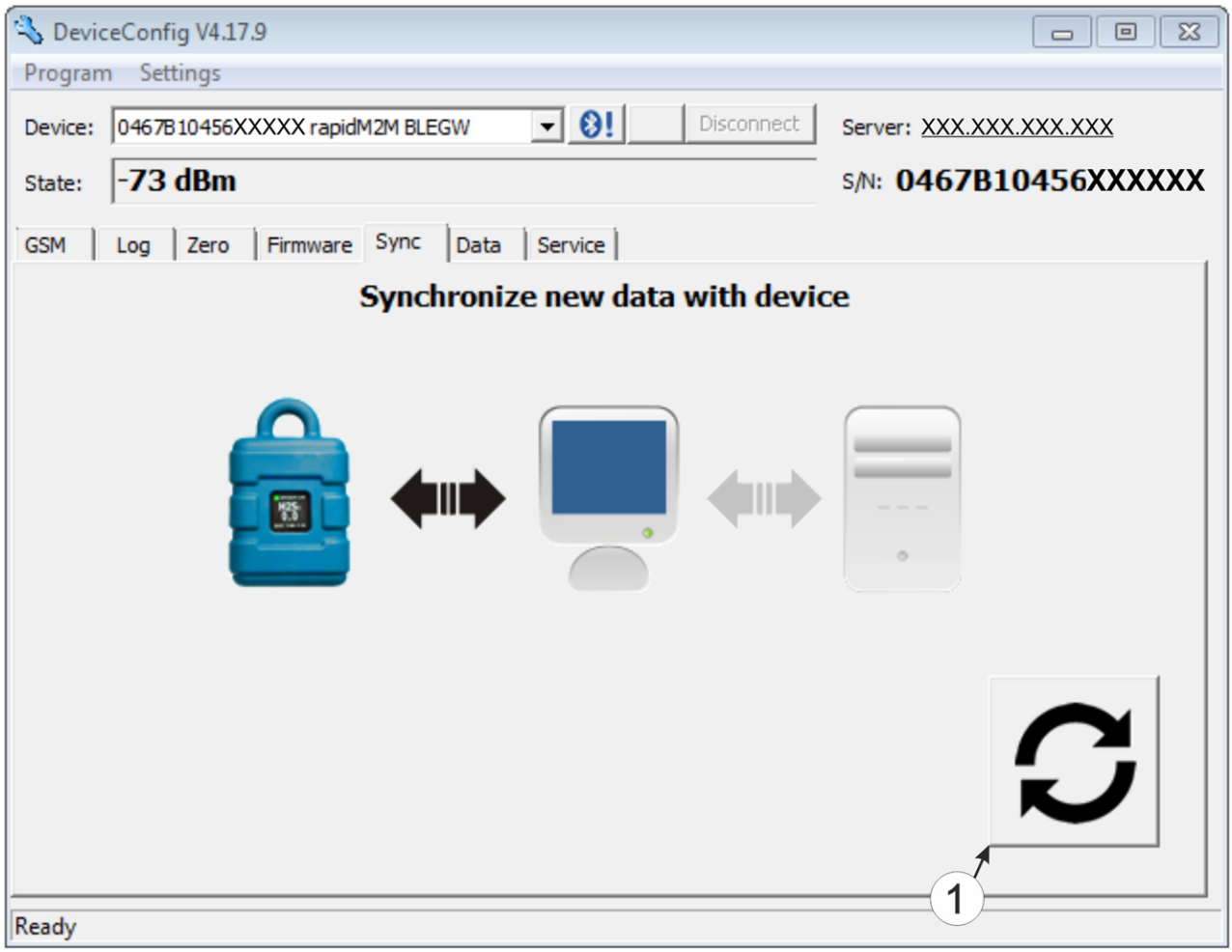
DeviceConfig

3. Connect the myDatalogEASY IoT GW ATEX to the PC using the USB BLE-Adapter (300685) supplied (see "Connecting a Device via Bluetooth Low Energy" on page 98).
4. Additional tabs are displayed if the connection was established successfully. Now select the "Sync" tab.



myDatalogEASY IoT GW ATEX specific tab

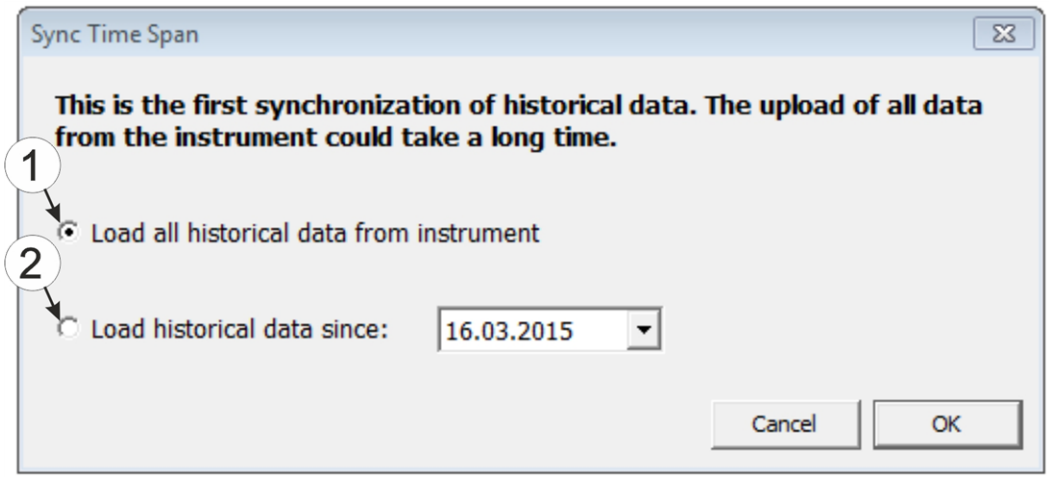
5. Click on the button to trigger synchronisation.



"Sync" tab when connected to the myDatalogEASY IoT GW ATEX although there is no connection to the myDatanet server

1	Button to trigger synchronisation

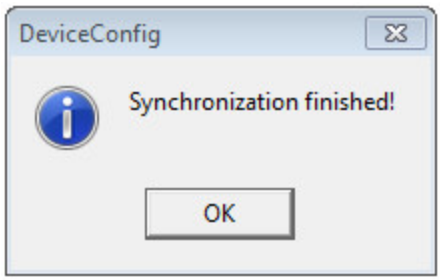
When you read out the data for the first time from a myDatalogEASY IoT GW ATEX , you can choose whether all of the saved data or only the data up to a certain date are read out from the myDatalogEASY IoT GW ATEX . During the following synchronisation processes, the DeviceConfig configuration program always reads out the data from the last synchronised measurement data record.



Selecting the time period for which the data should be read out (only during first synchronisation)

1	Read out all of the saved data ***Note:*** *Reading out all of the saved data can take several hours depending on the number of saved measurement data records.*
2	Only read out the data from the selected date onwards. The data is always read out from 00:00 am of the selected day. ***Important note:*** *Following completion of the synchronisation it is no longer possible to read out data before the selected date.*

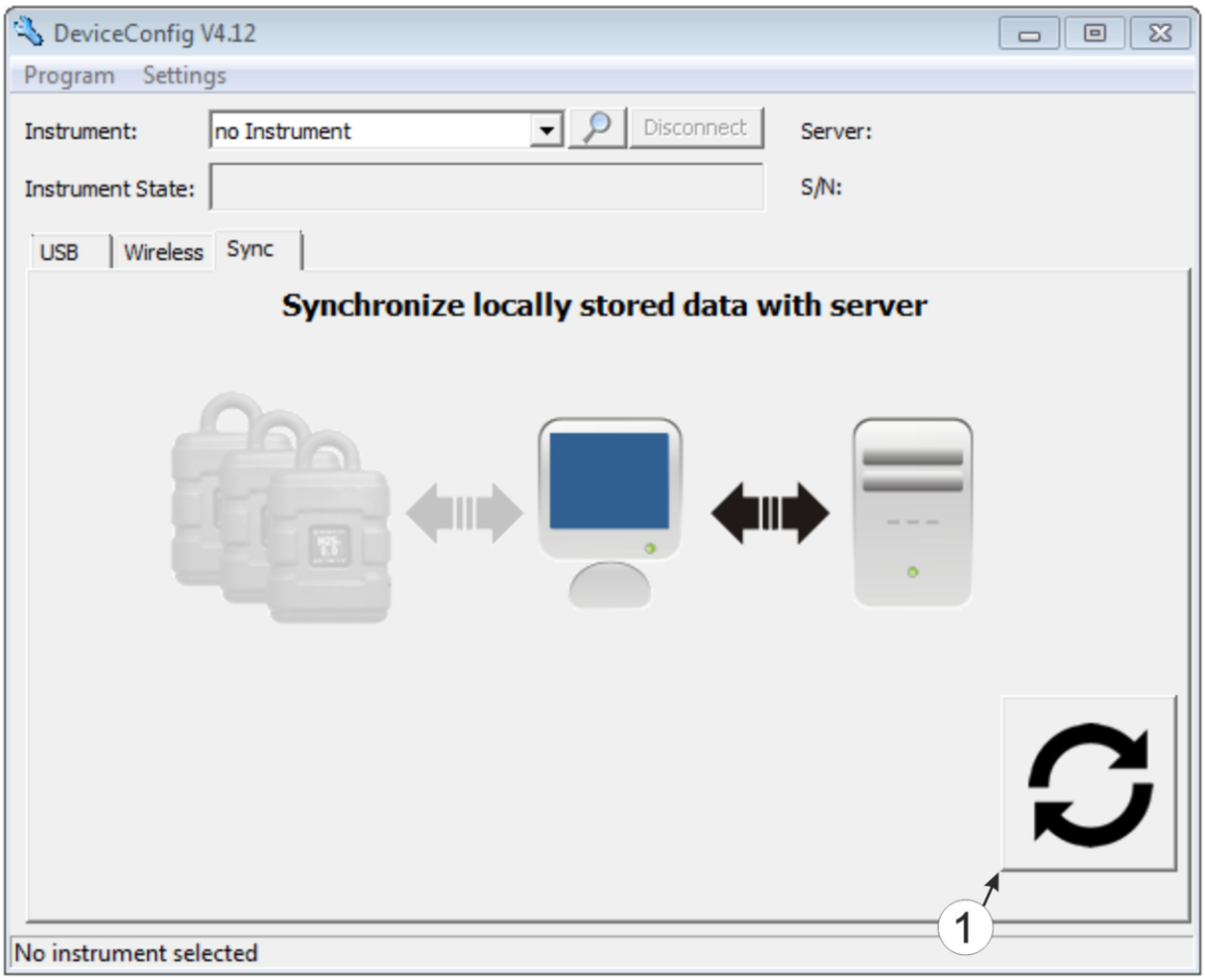
6. Wait until the DeviceConfig configuration program indicates that the synchronisation process is complete.



Synchronisation completed

7. Close the DeviceConfig configuration program.
8. Re-open the DeviceConfig configuration program as soon as your PC is connected to the Internet.

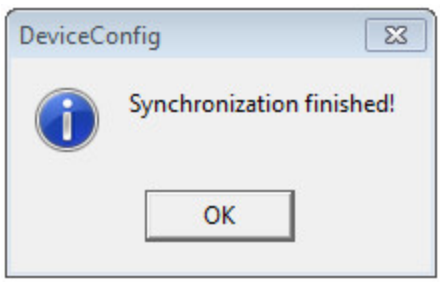
9. Select the "Sync" tab and click on the button to trigger synchronisation.



"Sync" tab without connection to a device

1	Button to trigger synchronisation During this process, the measurement data and configurations of all of the devices that the DeviceConfig configuration program has saved locally are synchronised with the myDatanet server.

10. Wait until the DeviceConfig configuration program indicates that the synchronisation process is complete.



Synchronisation completed

Chapter 12 "tbd" smartphone app

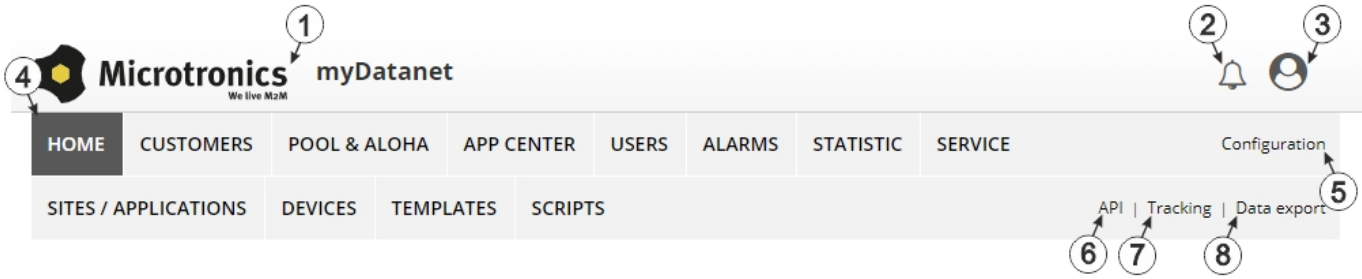
12.1 General

The "tbd" smartphone app is available for Android and iOS devices and can be downloaded free of charge from "Google Play" (Android) or the Apple "App Store" (iOS).

Chapter 13 myDatanet server

Note: *All of the screenshots show version 50v007 of the myDatanet server using the standard colour scheme. Newer versions may include minor changes to the appearance of the server.*

13.1 Overview



Overview of the myDatanet server

1	Freely selectable logo	**5**	Opens the screen to input the global settings for the server
2	Opens the window in which the notifications created by the system and intended for the currently logged-in user are summarized	**6**	Opens the rapidM2M Playground
3	Displays the menu for adjusting the user settings and for logging out the currently active user	**7**	Switches to the "Data exports" area to configure the data export. This tab is only visible if at least the licence for one export variant is available.
4	Tabs to switch between the individual server areas	**8**	Opens the input screen to upload a XML file. This tab is only visible if the licence for the XML import is available.

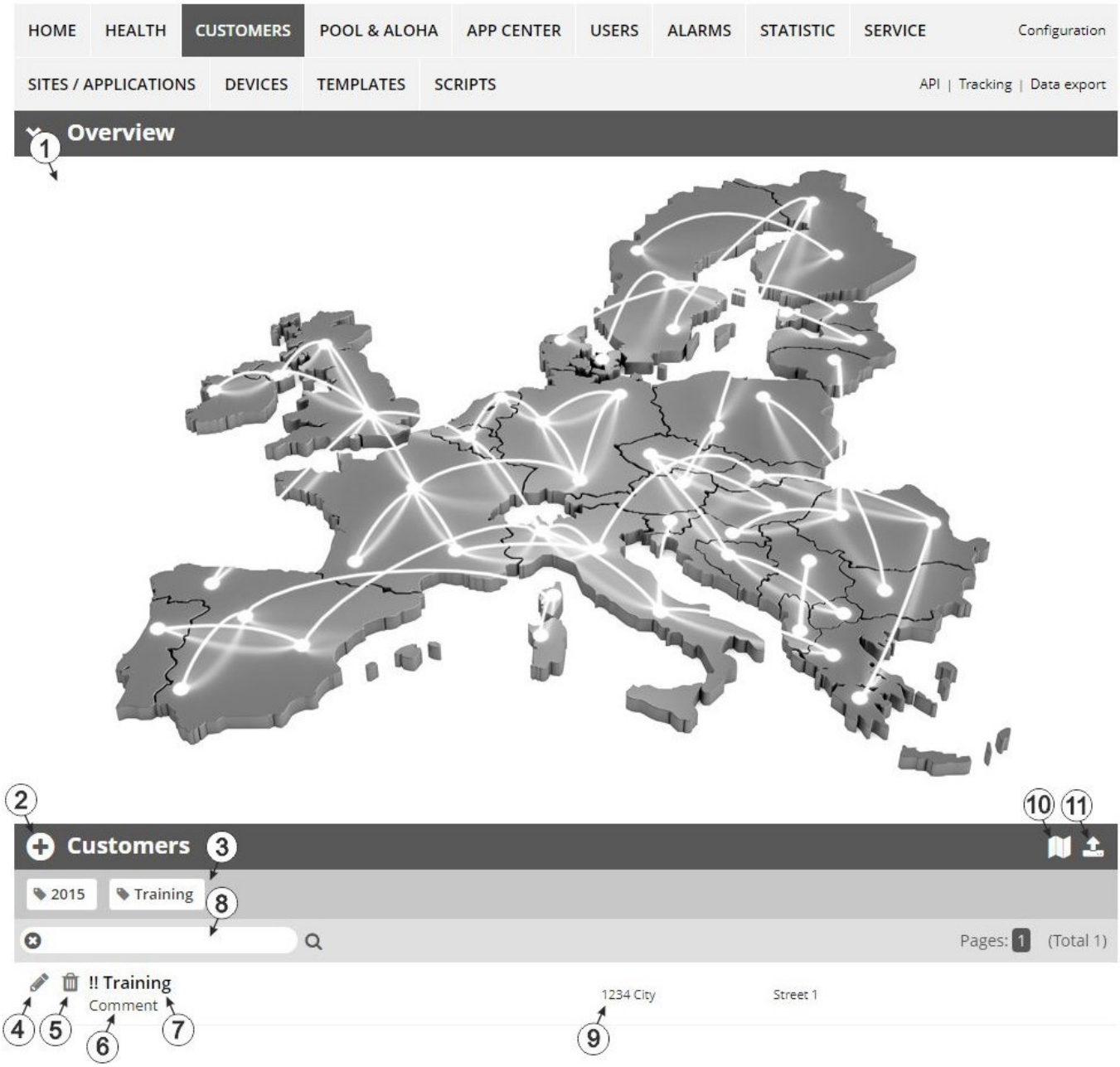
13.1.1 Explanation of the symbols


Adds a new entry to the current list (reports, sites, users, etc.).


Deletes the adjacent element (report, site, user, etc.) from the list.


Calls up the input screen to edit the adjacent element (report, site, user, etc.).

13.2 "Customer" area

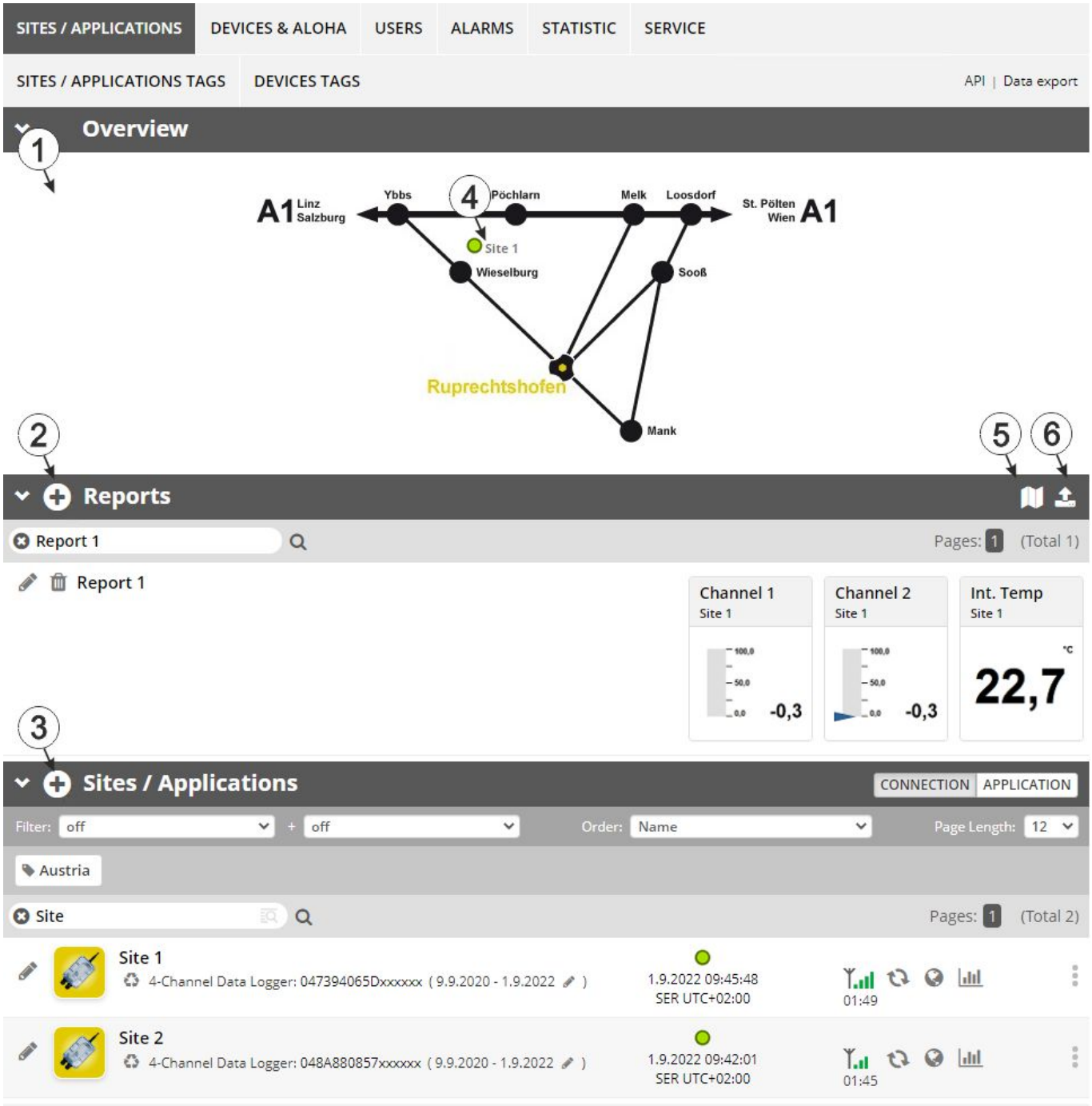


Overview of the "Customer" area

1	Area where an image file can be displayed as a "Map" and/or the OpenStreetMaps map can be displayed The sites can be manually placed on the image file used as a "map". In the OpenStreetMaps map, the sites are only displayed once GPS coordinates have been assigned to the site.
2	Adds a new customer

3	List of tags that are assigned to at least one of the customers displayed in the list of customers. If the list of customers was limited by the search field or selection of a tag, this is taken into consideration when creating the list of tags. A cross is added to the end of the list of tags as soon as the list of customers is limited by the selection of a tag. Clicking on this cross will reset the selection of all tags and the restriction is cancelled. By clicking on one of the tags with the left mouse button only those customers who have been assigned the corresponding tag are displayed in the list of customers and the selected tag is highlighted in colour. By clicking on one of the tags with the right mouse button all of the customers who have been assigned the corresponding tag are hidden, the selected tag is highlighted in colour and the title of the tag is crossed out. Clicking the same mouse button again will remove the restriction.
4	Opens the input screen for configuring the customer
5	Deletes the customer
6	Comment that can be entered in the configuration of the customer
7	If a default report was defined, the default report is accessed by clicking on the name of the customer. Otherwise the "Sites / Applications" area at customer level is opened by clicking on the name of the customer (see ""Sites / Applications" area at customer level" on page 120 or "Reports" on page 121).
8	Search field to filter the customer list
9	Customer's address that can be entered via the input screen for configuring the customer
10	Symbol via which a OpenStreetMaps map, on which the sites are displayed, can be loaded. (see "Map view" on page 121)
11	Symbol via which an image file can be loaded on to the server as an "Overview map" To remove the "Map" again, open the upload dialogue again and click on "Submit" without selecting an image file beforehand.

13.3 "Sites / Applications" area at customer level



Overview of the "Sites / Applications" area at customer level

1	Area where an image file can be displayed as a "Map" and/or the OpenStreetMaps map can be displayed The sites can be manually placed on the image file used as a "map". In the OpenStreetMaps map, the sites are only displayed once GPS coordinates have been assigned to the site.

2	List of reports (see "Reports" on page 121)
3	List of sites/applications (see "Site" on page 84)
4	Symbol that represents a site on the "Map"
5	Symbol via which a OpenStreetMaps map, on which the sites are displayed, can be loaded. (see "Map view" on page 121)
6	Symbol via which an image file can be loaded on to the server as a "Map" To remove the "Map" again, open the upload dialogue again and click on "Submit" without selecting an image file beforehand.

13.3.1 Reports

The reports provide a variety of options to display graphs of the data on the web interface of the myDatanet server or to download the data from the myDatanet server. Detailed instructions on creating and handling the reports is provided in myDatanet Server Manual (805002).

13.3.2 Map view

The map view provides an overview of the geographic position of the sites. Detailed instructions on operating and configuring map view are provided in myDatanet Server Manual (805002).

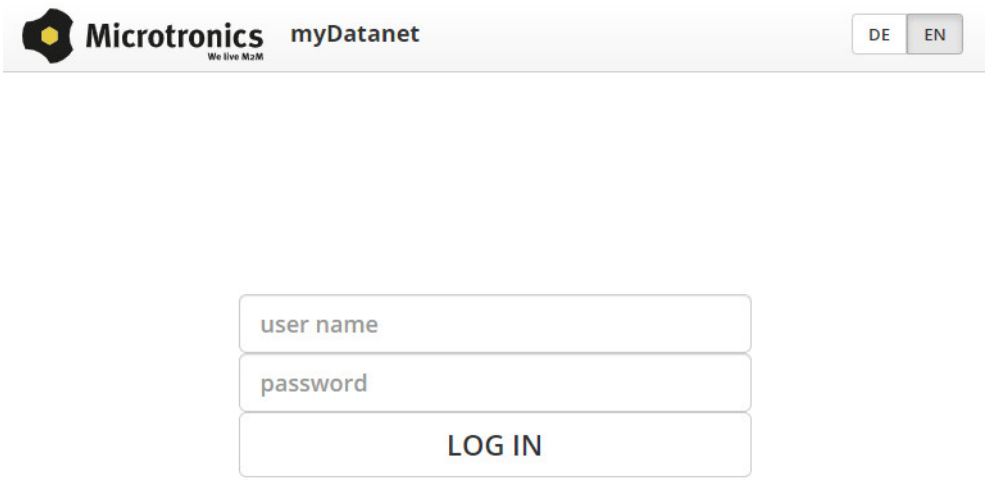
13.4 Recommended procedure

13.4.1 Creating the site

> ***Note:*** *Some of the fields mentioned in the following chapters may be hidden depending on the respective user level. In this case, please contact the administrator of the myDatanet server.*

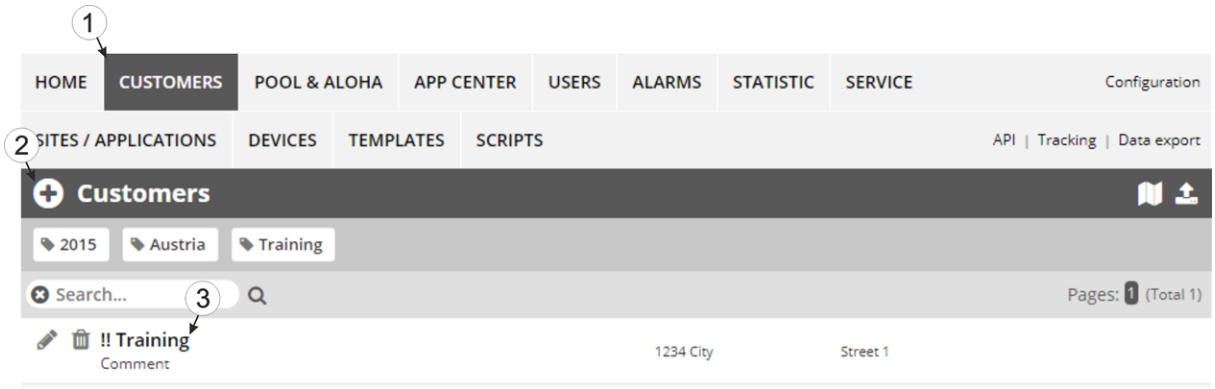
Detailed instructions on creating a new site are provided in myDatanet Server Manual (805002).

1. Log in via the web interface on the myDatanet server. You will receive the web address from your responsible sales partner.



Login form of the myDatanet server

2. Click on the "Customer" menu item of the myDatanet server to call up the list of available customers. Select an existing customer or create a new customer.

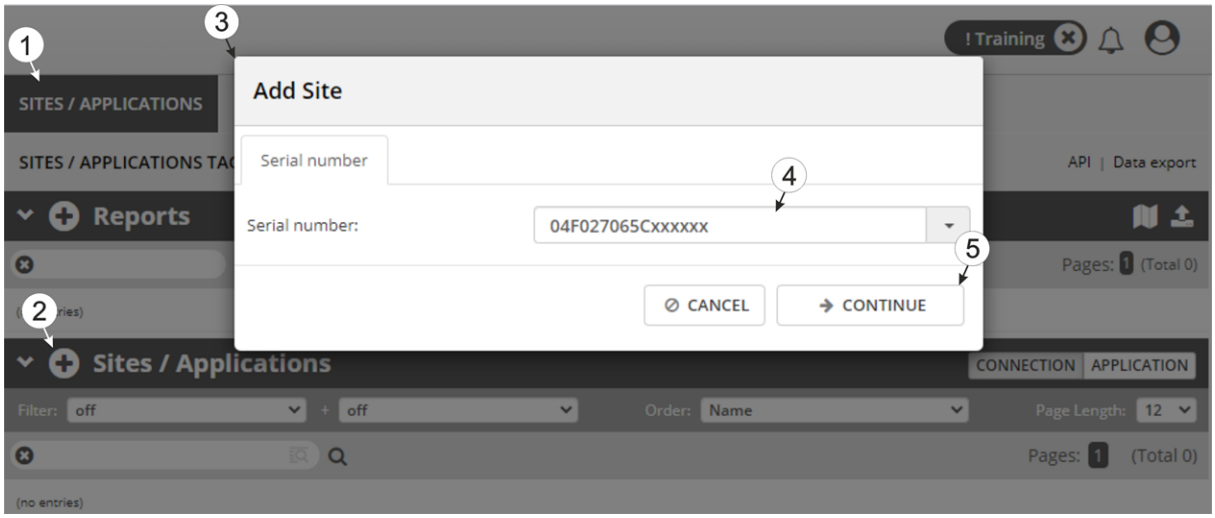


Selecting the customer

1 Menu item to call up the list of customers	**3** List of available customers
2 Creating a new customer	

3. Click on the "Sites / Applications" menu item of the myDatanet server to call up the list of existing sites / applications. Open the input window for creating a new site by clicking the "Add new site / application" symbol, enter the serial number of your device in the appropriate field and then click the "Continue" button.

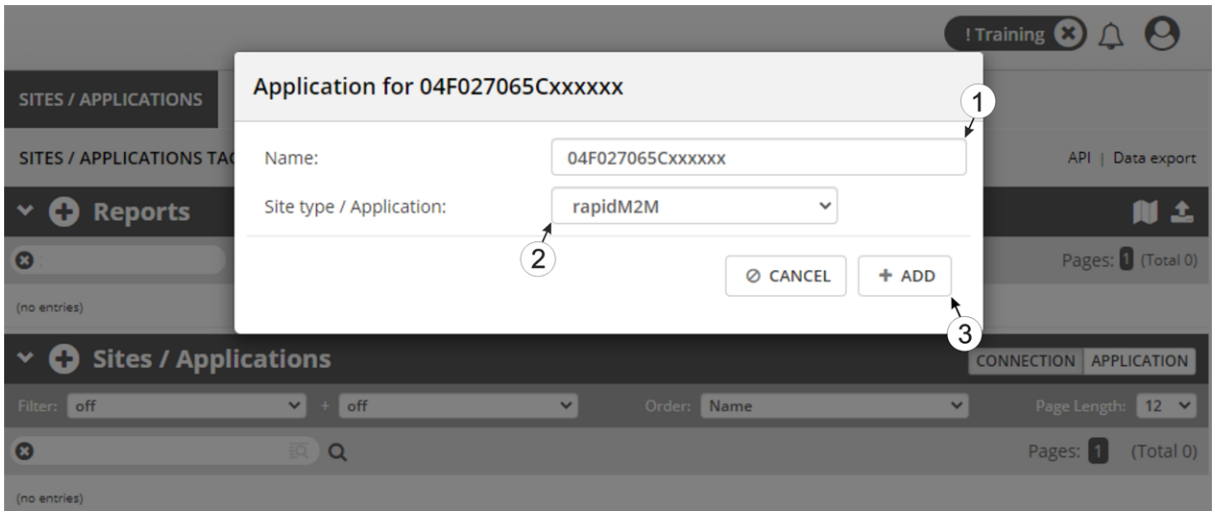
Note: *The serial number is on the type plate of the device (see "Device labelling" on page 32)*



Creating the site

1 Menu item to call up the list of existing sites / applications	**4** Field for entering the serial number
2 "Add new site / application" symbol	**5** "Continue" button
3 Input window for creating a new site	

4. If necessary, change the suggested name of the site, select the desired site type or the desired application from the drop-down list and then click the "Add" button.



Completing site creation

1 Name of the site (freely selectable)	**3** "Add" button
2 Drop-down list of available applications, templates and site types	

Chapter 14 rapidM2M Studio

Note: *The web-based development environment rapidM2M Studio is being developed continuously which can lead to slight changes to the appearance of the program compared to the screenshots used in this manual.*

14.1 General

Access to the web-based development environment rapidM2M Studio is included in the Microtronics Partner Program, for which you can register free of charge at the following address:

https://partner.microtronics.com

It is a web-based IDE that is designed to support customers with the creation of IoT applications for the myDatalogEASY IoT GW ATEX . This covers the entire development process - from editing the source code, to testing as part of the creation process to publishing the finished IoT application in the rapidM2M Store . All elements which make up an IoT application are summarised in a single project. The elements are:

- **Device logic:** intelligence installed locally on the myDatalogEASY IoT GW ATEX
- **Backend logic:** intelligence installed on the myDatanet server
- **Data descriptor:** describes the structure of the data (measurement data, configurations, etc.), that is exchanged between myDatalogEASY IoT GW ATEX , myDatanet server and external systems (e.g. front ends connected via REST API).
- **Portal view:** Simple front end that is supplied by the myDatanet server (e.g. for fast prototype development and/or provision of administrative data)

In addition to the dashboard (see "Project dashboard" on page 127) for managing the projects, the rapidM2M Studio consists of two main interfaces:

- **CODEbed:** Editing and compiling the source codes (see "CODEbed" on page 128)
- **TESTbed:** Testing the IoT application in conjunction with a locally connected device and the associated back end i.e. the myDatanet server (see "TESTbed" on page 129)

The sharing function implemented in the rapidM2M Studio enables developers from different disciplines (firmware programmers, cloud developers, web designers, etc.) to create an IoT application together as well as to share projects and libraries with colleagues and the community. The integrated version management also ensures controlled distribution of updates of an IoT application across the entire chain from the rapidM2M Studio to the rapidM2M Store to the sites (that were created based on the IoT application) to the myDatalogEASY IoT GW ATEX .

14.2 Prerequisites

Interfaces	1 x USB
Operating system	Windows 7 Windows 10 (recommended) MacOS 10.12 or higher Linux (Fedora 32, Ubuntu 20.04, Archlinux 2020.06.01)
Internet connection	Required
Required disk space	No installation required
Browser	Google Chrome only

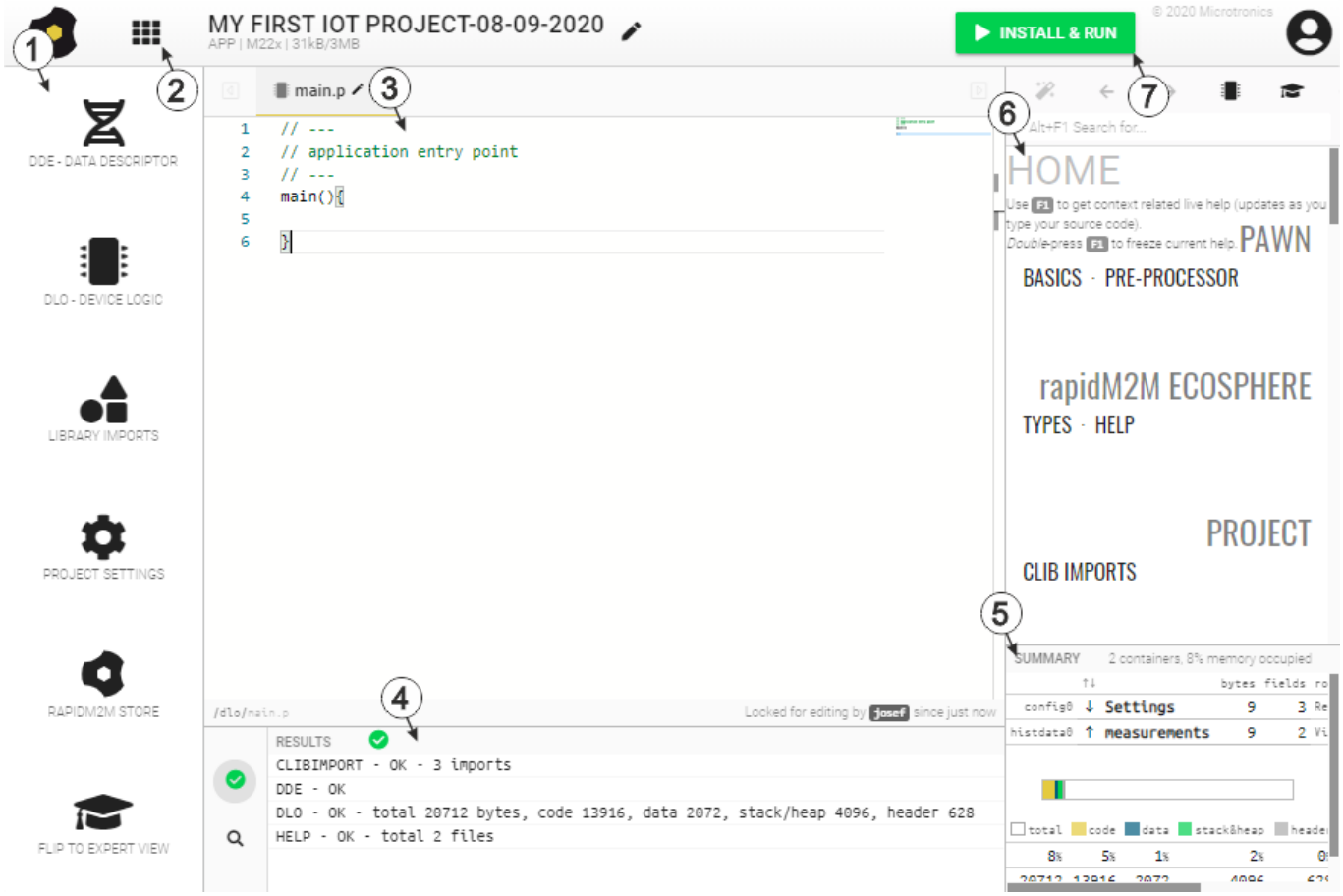
14.3 Project dashboard



Project dashboard of the rapidM2M Studio

1	Search field to filter the list of projects	
2	Button for switching sorting of the project list according to alphabetical order or last use	
3	Opens the quick guide for the rapidM2M Studio	
4	Button for displaying the menu that contains all relevant settings for the currently active user	
5	Button for creating a new project	
6	Buttons for filtering the list of the projects according to:	
		Recently used
		All my projects
		Projects shared by me
		Projects shared with me
7	Tile that contains all important information about an IoT project	
8	List of the "Collections"	
		All projects that are not assigned to another "Collection"
		Favourite projects
		Sample libraries provided by Microtronics
		Samples provided by Microtronics
		"Collection" created by user
9	Button for creating a new "Collection"	

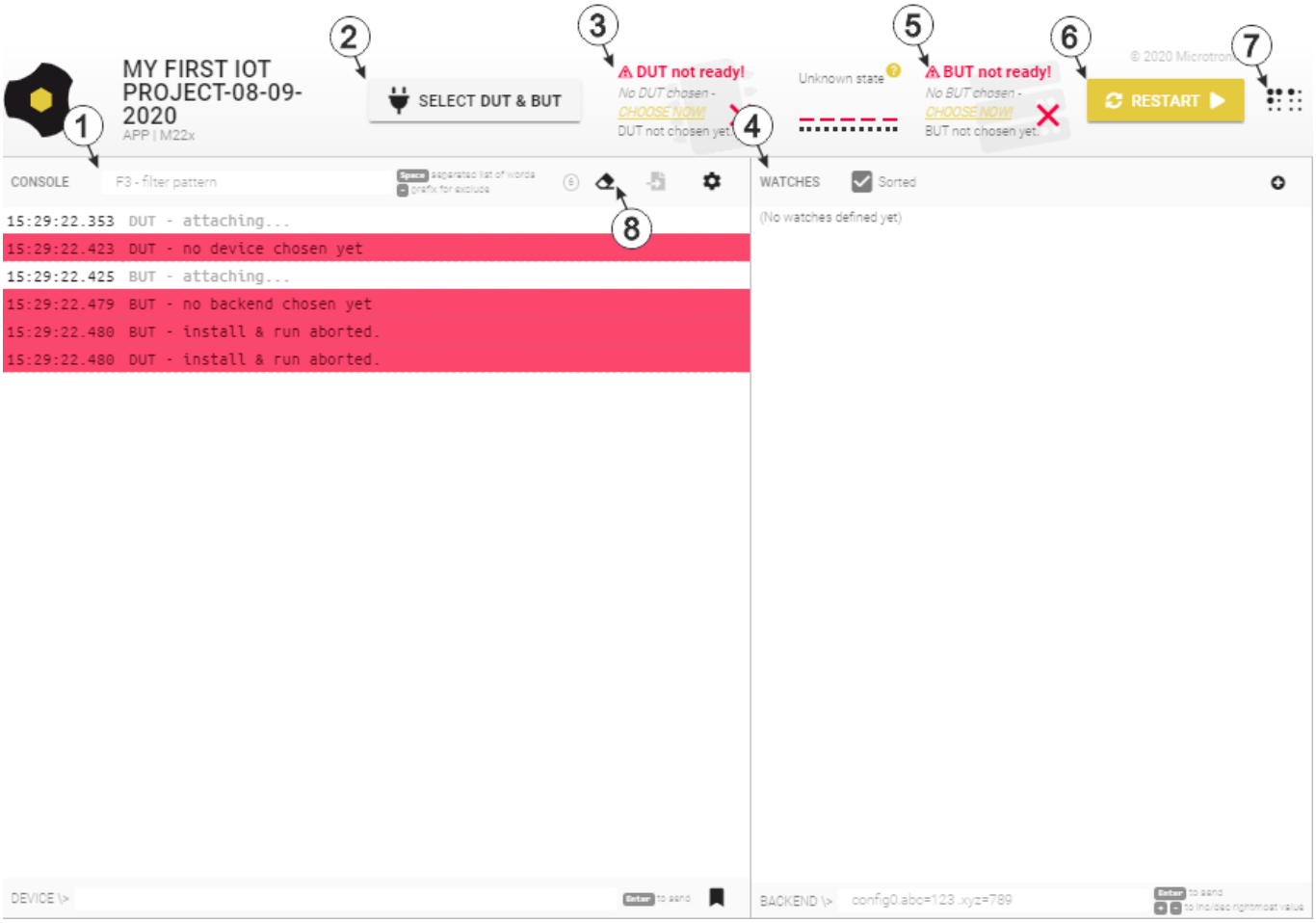
14.4 CODEbed



CODEbed of the rapidM2M Studio

1	Navigation panel
2	Back to the project dashboard
3	Editor panel
4	Compiler results incl. warnings and errors
5	Memory usage
6	Context-sensitive help
7	Installs the current binaries of the project on the device and backend (i.e. on the myDatanet server) and opens the TESTbed

14.5 TESTbed



TESTbed of the rapidM2M Studio

1	Debug console
2	First opens the window for selecting and connecting the "Device under test" and then the window for entering the access data for the "Backend under test"
3	Information on the "Device under test"
4	Watch panel
5	Information on the "Backend under test" (i.e. the myDatanet server)
6	Restarts the device logic installed on the device. The device logic is reloaded onto the device for this purpose. However, any changes made in the CODEbed are not taken into account here. This is only done again by clicking the button "Install & Run" in the CODEbed.
7	Button for displaying/fading out additional panels
8	Button for deleting the console output

Chapter 15 Device Logic

15.1 General

The following chapter describes the functionality of the device logic. The programming language used is built on Pawn, a scripting language similar to C that runs on embedded systems.

Additional, more detailed information is provided on the developer's website:
http://www.compuphase.com/pawn/pawn.htm.

There are several ways to create a device logic for the myDatalogEASY IoT GW ATEX :

- Direct entry in the "Device Logic" input field in the "Control" configuration section
- Upload of a previously created binary file (*.amx) to the myDatanet server
- Usage of the CODEbed (see "CODEbed" on page 128) of the web-based development environment rapidM2M Studio

15.1.1 Direct entry of a device logic

The device logic is entered via the "Control" configuration section (see "Control" on page 85) of the input screen for configuring the site. "Pawn" must be selected as the "Device Logic Type" so that the myDatalogEASY IoT GW ATEX interprets the commands entered under "Device Logic" as a script.

15.1.2 Uploading a binary file

If the "Upload a compiled device logic" entry was selected via the "Device logic source" list selection in the "Control" configuration section (see "Control" on page 85) of the input screen for configuring the site, a binary file that was, for example, previously created via the web-based development environment rapidM2M Studio (see "rapidM2M Studio " on page 125) can be uploaded to the myDatanet server. This is then loaded into the myDatalogEASY IoT GW ATEX during the next connection. When using this method, "Pawn" must also be selected as the "Device Logic Type" so that the myDatalogEASY IoT GW ATEX interprets the commands as a script.

15.1.3 Using the CODEbed of the web-based development environment rapidM2M Studio

The CODEbed is one of the two main interfaces of the web-based development environment rapidM2M Studio . The CODEbed serves to create and compile source codes for all elements (device logic, backend logic, data descriptor and portal view) of an IoT application. The functional scope of the rapidM2M Studio also includes transfer of the compiled device logic into the myDatalogEASY IoT GW ATEX via a USB connection and copying of the data descriptor to the development site with which the myDatalogEASY IoT GW ATEX is linked.

15.2 Device API

15.2.1 Constants

Return codes for general purposes

```
OK                        = 0,
ERROR                     = -1,
ERROR_PARAM               = -2,  // Parameter error
ERROR_UNKNOWN_HDL         = -3,  // Unknown handler, handle or resource error
ERROR_ALREADY_SUBSCRIBED  = -4,  // Already subscribed service or resource error
ERROR_NOT_SUBSCRIBED      = -5,  // Not subscribed service error
ERROR_FATAL               = -6,  // Fatal error
ERROR_BAD_HDL             = -7,  // Bad handle or resource error
ERROR_BAD_STATE           = -8,  // Bad state error
ERROR_PIN_KO              = -9,  // Bad PIN state error
ERROR_NO_MORE_HANDLES     = -10, /* The service subscription maximum capacity is
                                    reached */
ERROR_DONE                = -11, /* The required iterative process is now
                                    terminated */
ERROR_OVERFLOW            = -12, /* The required operation has exceeded the
                                    function capabilities */
ERROR_NOT_SUPPORTED       = -13, /* An option, required by the function, is not
                                    enabled on the CPU, the function is not
                                    supported in this configuration */
ERROR_NO_MORE_TIMERS      = -14, /* The function requires a timer subscription,
                                    but no more timer resources are available */
ERROR_NO_MORE_SEMAPHORES  = -15, /* The function requires a semaphore allocation,
                                    but there are no more semaphore resources */
ERROR_SERVICE_LOCKED      = -16, /* The function was called from a low or high
                                    level interrupt handler (the function is
                                    forbidden in this case) */
ERROR_MEM                 = -100,// error allocating memory
ERROR_SIM_STATE           = -101,// SIM state error
ERROR_MODEM_DISABLED      = -102,// Modem disabled
ERROR_SENSOR_DISABLED     = -102,/* Sensor disabled
                                    (Alias for ERROR_MODEM_DISABLED)*/
ERROR_FEATURE_LOCKED      = -103,// feature locked
ERROR_TXITF               = -104,/* tx interface (uplink) not available
                                    (e.g. not opened, currenly closing) */
```

15.2.2 Timer, date & time

15.2.2.1 Arrays with symbolic indices

TrM2M_DateTime

Detailed breakdown of the date and time

```
// year         Year specified relates to the 21st century, i.e. 14 refers to
//              the year 2014
// month        Month   (1..12)
// day          Day     (1..31)
// hour         Hours   (0..23)
// minute       Minutes (0..59)
// second       Seconds (0..59)
// DoW          Weekday (0 = Monday ... 6 = Sunday)
// timestamp    Time stamp (seconds since 31.12.1999)
// timestamp256 Fraction of the next started sec. (resolutions 1/256 sec.)

#define TrM2M_DateTime[ .year, .month, .day, .hour, .minute, .second, .DoW,
                        .timestamp, .timestamp256                          ]
```

15.2.2.2 Constants

Time basis flags

Control flags for the rM2M_SetDateTime() function

```
RM2M_DATETIME_LOCALTIME = 0b00000001, // transferred time in local time
```

15.2.2.3 Functions

native rM2M_GetTime(&hour=0, &minute=0, &second=0, timestamp=0);

If no time stamp was transferred (timestamp=0), the current system time (in UTC) is converted to hours/minutes/seconds. Alternatively, the transferred time stamp is converted to hours/minutes/seconds.

Parameter	*Explanation*
hour	*Variable to store the hours - OPTIONAL*
minute	*Variable to store the minutes - OPTIONAL*
second	*Variable to store the seconds - OPTIONAL*
timestamp	*Time stamp that should be converted* *= 0: The current system time (in UTC) is converted.* *> 0: The transferred time stamp is converted.* *(The time stamp must be specified in seconds since 31.12.1999.)*

	Explanation
Return value	• *timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)* • *timestamp > 0: The transferred time stamp is returned.*

native rM2M_GetDate(&year=0, &month=0, &day=0, timestamp=0);

If no time stamp was transferred (timestamp=0), the date (year, month, day) is determined for the current system time (in UTC). Alternatively, the date (year, month, day) is determined for the transferred time stamp.

Parameter	***Explanation***
year	*Variable to store the year - OPTIONAL* ***Note:*** *The year specified relates to the 21st century, i.e. the value 14 refers to the year 2014.*
month	*Variable to store the month - OPTIONAL*
day	*Variable to store the day - OPTIONAL*
timestamp	*Time stamp for which the date should be determined* *= 0: The date for the current system time (in UTC) is determined.* *> 0: The date for the transferred time stamp is determined.* *(The time stamp must be specified in seconds since 31.12.1999.)*

	Explanation
Return value	• *timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)* • *timestamp > 0: The transferred time stamp is returned.*

native rM2M_GetDateTime(datetime[TrM2M_DateTime]);

Reads the current time (in UTC) and date from the system

Parameter	***Explanation***
datetime	*Structure for storing a detailed breakdown of the date and time (see "TrM2M_ DateTime" in chapter "Arrays with symbolic indices" on page 133)*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native rM2M_SetDateTime(datetime[TrM2M_DateTime], flags=0);

Sets the system date and time to the values contained in the transferred structure

Parameter	*Explanation*
datetime	*Structure that contains a detailed breakdown of the date and time (see "TrM2M_DateTime" in chapter "Arrays with symbolic indices" on page 133).* *.timestamp = 0:* *The values contained in .year, .month, .day, .hour, .minute and .second are used to set the date/time.* *.timestamp != 0:* *The time stamp contained in .timestamp is used to set the date/time.*
flags	*Configuration flags for setting the system time - OPTIONAL* *Bit0 (RM2M_DATETIME_LOCALTIME):* *must be set if the transferred structure contains the time in local time*

	Explanation
Return value	• *> 0, difference in seconds between the current time and time to be set* • *0, if the difference between the current time and time to be set is less than 5 sec.* • *ERROR, if invalid parameters were transferred* • *ERROR-1, if the time to be set is more than one day ahead of the current system time*

native rM2M_GetTimezoneOffset();

Returns the difference (in seconds) between the system time (UTC) and local time configured for the site on the myDatanet server. This can be used to determine the local time in the script by adding the difference to the system time (UTC). The offset value is determined by the myDatanet server in accordance with the set time zone (including summer/winter time) and is synchronised during every connection to the device.

Example: Central European time (CET = UTC+1) is used for the site -> Offset = 3600 sec.

	Explanation
Return value	*Offset value in seconds*

native rM2M_DoW(timestamp);

Calculates the weekday from a given timestamp

Parameter	*Explanation*
timestamp	*Timestamp of the day in question*

	Explanation
Return value	*Weekday, 0=Monday ... 6=Sunday*

native rM2M_TimerAdd(funcidx);

Generates a new 1s timer

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up following expiry of the timer* *Type of function: public func();*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, If one of the following errors occurs:*
	  • *No valid index was transferred*
	  • *No further timers can be created (maximum number reached)*
	  • *In the event of an internal error*
	• *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_TimerRemove(funcidx);

Removes a 1s timer

Parameter	*Explanation*
funcidx	*Index of the public function of the timer that should be removed* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index was transferred or in the event of an internal error* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_TimerAddExt(funcidx, bool:cyclic, time);

Generates a new ms timer

Important note: *The maximum number of simultaneously active ms timers is 8.*

Parameter	***Explanation***
funcidx	*Index of the public function that should be called up following expiry of the timer* *Type of function: public func();*
cyclic	*Setting for the behaviour following expiry of the timer interval:* *true: The timer must be restarted following expiry of the interval.* *false: The timer is stopped following expiry of the interval.*
time	*Timer interval in milliseconds (max. 60,000 ms)* ***Note:*** *By setting the interval to 0ms, a timer can be generated for which the call back function is called up immediately after the current code block (e.g. main function) is executed. However, only single shot timers (i.e. the timer is stopped upon expiry of an interval) may be initialised with an interval of 0ms.*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, if one of the following errors occurs*
	  • *No valid index was transferred.*
	  • *An interval of 0ms was specified and the timer should be restarted automatically upon expiry of the timeout (i.e. cyclical 0ms timer).*
	  • *Internal error*
	  • *No additional timers can be created (maximum number reached).*
	• *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_TimerRemoveExt(funcidx);

Removes a ms timer

Parameter	***Explanation***
funcidx	*Index of the public function of the timer that should be removed* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index was transferred or in the event of an internal error* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

15.2.3 Uplink

15.2.3.1 Arrays with symbolic indices

TrM2M_GSMInfo

Information regarding the GSM modem, SIM chip and the GSM network used during the last connection

```
// cgmi       Manufacturer identification of the modem
// cgmm       Modem model information
// cgmr       Modem revision information
// imei       International mobile equipment identity of the modem
// imsi       International mobile subscriber identity of the SIM chip that was
//            used for the last connection
//            Empty string, if no connection has been established yet
// iccid      Integrated circuit card identifier of the SIM chip that was used
//            for the last connection
//            Empty string, if no connection has been established yet
// mcc        MCC (Mobile Country Code) of the network used for the last/current
//            connection
//            0, if no connection has been established yet
// mnc        MNC (Mobile Network Code) of the network used for the last/current
//            connection
//            0, if no connection has been established yet
// simstate   Current SIM state (see "SIM state" in chapter
//            "Constants" on page 139)
// act        Radio access technology used for the last/current connection (see
//            "Mobile radio AcT" in chapter "Constants" on page 139)
// lac        LAC (location area code) of the network used for the last/current
//            connection
// cid        Cell identifier of the network used for the last/current
//            connection
//            2G AcT:16-bit cell ID
//            3G AcT:28-bit UTRAN cells ID (16-bit cells ID + 12-bit RNC-ID)
//            4G AcT:28-bit E-UTRAN cells ID (8-bit sector ID + 20-bit
//            eNodeB-ID)

#define TrM2M_GSMInfo[  .cgmi{20}, .cgmm{20}, .cgmr{20}, .imei{16}, .imsi{16},
                        .iccid{21}, .mcc, .mnc, .simstate, .act, .lac, .cid   ]
```

TrM2M_TxItfStats

Statistical information on the uplink communication interface

```
// rtt        Time [ms] it takes for the device to receive an answer from the
//            server for a keep alive ping sent to the server (round trip
//            time)1)

#define TrM2M_TxItfStats [.rtt]
```

1) can only be determined if the "Bidirectional alive ping" is activated on the server. The "Bidirectional alive ping" enables the device and server to easily detect whether the connection is still established. The "Bidirectional alive ping" can be activated globally for the complete server, for a specific customer or for a single site (see "myDatanet Server Manual " 805002).

15.2.3.2 Constants

SIM state

```
//Connection can be initiated via devic logic
RM2M_SIM_STATE_NONE       = 0,   //Initial state
RM2M_SIM_STATE_PRODUCTION = 1,   //Newly produced device is in stock
RM2M_SIM_STATE_HOT        = 2,   //Valid contract

//Connection cannot be initiated via device logic
RM2M_SIM_STATE_COLD       = 3,   /*End of the contract or fair use policy
                                   violated*/
RM2M_SIM_STATE_DISCARDED  = 4,   //Device has been decommissioned
```

Mobile radio AcT (access technology)

```
// Mobile radio AcT (access technology) as per 3GPP TS27.007
RM2M_TX_ACT_GSM                 = 0,   // GSM
RM2M_TX_ACT_GSM_COMPACT,        = 1,   // GSM compact
RM2M_TX_ACT_UTRAN,              = 2,   // UTRAN
RM2M_TX_ACT_GSM_W_EGPRS,        = 3,   // GSM with EGPRS
RM2M_TX_ACT_UTRAN_W_HSDPA,      = 4,   // UTRAN with HSDPA
RM2M_TX_ACT_UTRAN_W_HSUPA,      = 5,   // UTRAN with HSUPA
RM2M_TX_ACT_UTRAN_W_HSDPA_HSUPA = 6,   // UTRAN with HSDPA and HSUPA
RM2M_TX_ACT_E_UTRAN,            = 7,   // E-UTRAN

//rapidM2M specific
RM2M_TX_ACT_WIFI                = 100, // WiFi
RM2M_TX_ACT_ETH                 = 101, // Ethernet

RM2M_TX_ACT_UNKNOWN             = 255  // Unknown
```

Connection flags

Control flags for the rM2M_TxStart() function

```
RM2M_TX_POSUPDATE          = 0b00000001, /* Update of the GSM position data
                                            when establishing a connection    */
RM2M_TX_REFRESH_CONFIG     = 0b00000100, /* Additionally establishing a
                                            connection to the maintenance
                                            server                            */
RM2M_TX_SUPPRESS_POSUPDATE = 0b00001000, /* Suppress update of the GSM position
                                            data when establishing a
                                            connection [1)]                   */
RM2M_TX_POSUPDATE_ONLY     = 0b00010000, /* When establishing a connection, only
                                            the GSM position data is updated.
                                            Measurement data, configurations,
                                            etc. are not synchronised.        */
```

1) This suppresses the update of the GSM position data that is automatically executed by the firmware every 24h .

Communication modes

Communication modes for the rM2M_TxSetMode() function

```
RM2M_TXMODE_TRIG   = 0,          // Interval
RM2M_TXMODE_WAKEUP = 1,          // Interval & wakeup
RM2M_TXMODE_ONLINE = 2,          // Online
```

Communication mode flags

Configuration flags for the rM2M_TxSetMode() function

```
RM2M_TXMODE_SUPPRESS_SYNC = 0b00000001, /* no auto. sync. with the server when
                                           the connection type is changed */
```

Connection status

Return values of the rM2M_TxGetStatus() function

```
RM2M_TX_FAILED     = 0b0000000001,       // Connection establishment failed
RM2M_TX_ACTIVE     = 0b0000000010,       // GPRS connection established
RM2M_TX_STARTED    = 0b0000000100,       // Connection establishment started
RM2M_TX_RETRY      = 0b0000001000,       // Delay until retry
RM2M_TX_WAKEUPABLE = 0b0000010000,       // Modem is logged into the GSM network
RM2M_TX_EXTSIM     = 0b0000100000,       // external SIM is used
RM2M_TX_DISABLED   = 0b0001000000,       // Modem was deactivated
RM2M_TX_WAKEUP     = 0b0100000000,       /* Connection establishment triggered
                                            by wakeup SMS */
RM2M_TX_POSUPDATE_ACTIVE = 0b1000000000, // Positioning is running
```

Connection error codes

Error codes that are returned by the rM2M_TxGetStatus() function via the optional "errorcode" parameter if the last connection attempt failed.

```
RM2M_TXERR_NONE = 0,              // no error

// general errors
RM2M_TXERR_CONNECTION_TIMEOUT,    // connection timed out
RM2M_TXERR_NEWDATA_TIMEOUT,       // timeout during server sync in online mode
RM2M_TXERR_IRREGULAR_OFF,         // irregularly closed connection
RM2M_TXERR_SERVER_NOT_AVAILABLE,  // server not available
RM2M_TXERR_SERVER_COMMUNICATION,  // error during communication with server

// general modem errors
RM2M_TXERR_MODEM = 10,            // unspecified modem error
RM2M_TXERR_MODEM_TIMEOUT,         // timeout modem communication
RM2M_TXERR_MODEM_HW_NOT_FOUND,    // modem not found
RM2M_TXERR_MODEM_HW_UNKNOWN,      // unknown modem
RM2M_TXERR_MODEM_INIT,            // error during init
RM2M_TXERR_MODEM_UNS_RESTART,     /* unsolicited restart (e.g. due to weak power
                                     supply) */

// SIM related errors
RM2M_TXERR_MODEM_SIM = 30,        // unspecified SIM related error
RM2M_TXERR_MODEM_SIM_NO_ATTEMPT,  // only one remaining pin input attempt
RM2M_TXERR_MODEM_SIM_PIN_WRONG,   // pin code is wrong
```

```
RM2M_TXERR_MODEM_SIM_NO_PIN,      // pin code required but not available
RM2M_TXERR_MODEM_EXTSIM_DENIED,   /* external SIM not allowed (APN and/or
                                     feature key) */
RM2M_TXERR_MODEM_EXTSIM_MISSING,  // external SIM not found
RM2M_TXERR_MODEM_SIM_OTHER,       /* any other problem with SIM card (e.g.
                                     internal SIM not found) */

// network-related error (GSM, GPRS, PDP, etc.)
RM2M_TXERR_MODEM_NETWORK = 50,    // unspecified network related error
RM2M_TXERR_MODEM_GSM_BAND_SEL,    // GSM not available (e.g. error with antenna)
RM2M_TXERR_MODEM_NETLOCK,         /* error registering within network (e.g. not
                                     allowed) */
RM2M_TXERR_MODEM_POSUPDATE,       // error with GSM position update
RM2M_TXERR_MODEM_PDP_CTX,         // error activating PDP context

// TCP related modem errors
RM2M_TXERR_MODEM_TCP = 70,        /* TCP error (e.g. timeout, server not
                                     available) */

// general WIFI errors
RM2M_TXERR_WIFI = 200,            // unspecified WIFI error
RM2M_TXERR_WIFI_TIMEOUT,          // timeout WIFI communication
RM2M_TXERR_WIFI_HW_NOT_FOUND,     // WIFI device not found
RM2M_TXERR_WIFI_INIT,             // error during init
RM2M_TXERR_WIFI_IO,               // error IO communication

// network-related WIFI errors
RM2M_TXERR_WIFI_NETWORK = 220,    // unspecified network related WIFI error
RM2M_TXERR_WIFI_NETWORK_TIMEOUT,  // timeout accessing network
RM2M_TXERR_WIFI_AP_SCAN_TIMEOUT,  /* timeout scanning for available access
                                     points */
RM2M_TXERR_WIFI_AP_SCAN,          /* error scanning access points (e.g.
                                     currently not possible) */
RM2M_TXERR_WIFI_DHCP_TIMEOUT,     /* timeout receiving IP address from DHCP
                                     server */
RM2M_TXERR_WIFI_AP_SETTINGS,      // access point settings not plausible
RM2M_TXERR_WIFI_AP_CONNECT,       // error connecting to access point
RM2M_TXERR_WIFI_AP_NOT_FOUND,     // access point not found during scan

// TCP related WIFI errors
RM2M_TXERR_WIFI_TCP = 240,        // unspecified TCP related WIFI error
RM2M_TXERR_WIFI_TCP_OPEN_TO,      // timeout opening TCP connection
RM2M_TXERR_WIFI_TCP_SEND_TO,      // timeout sending data
RM2M_TXERR_WIFI_TCP_CONNECT,      // error connecting to server
RM2M_TXERR_WIFI_TCP_FAILED,       // other error concerning TCP connection

// general Ethernet errors
RM2M_TXERR_ETH = 300,             // unspecified Ethernet error
RM2M_TXERR_ETH_TIMEOUT,           // timeout Ethernet communication
RM2M_TXERR_ETH_INIT,              // error during init
RM2M_TXERR_ETH_IO,                // error IO communication
RM2M_TXERR_ETH_INIT_MAC_PHY,      // error initialising MAC/PHY interface
RM2M_TXERR_ETH_ITF_UP,            /* TCP/IP stack: error bringing itf up
                                     (including dhcp) */
```

```
// network-related Ethernet errors
RM2M_TXERR_ETH_NETWORK = 320,     // unspecified network related Ethernet error
RM2M_TXERR_ETH_NETWORK_TIMEOUT,   // timeout accessing network
RM2M_TXERR_ETH_DHCP_TIMEOUT,      /* timeout receiving IP address from DHCP
                                     server */

// TCP-related Ethernet errors
RM2M_TXERR_ETH_TCP = 340,         // unspecified TCP related Ethernet error
RM2M_TXERR_ETH_TCP_OPEN_TIMEOUT,  // timeout opening TCP connection
RM2M_TXERR_ETH_TCP_SEND_TIMEOUT,  // timeout sending data
RM2M_TXERR_ETH_TCP_CONNECT,       // error connecting to server
RM2M_TXERR_ETH_TCP_FAILED,        // other error concerning TCP connection
```

Available uplink interfaces

Selectable uplink interfaces for the rM2M_TxSelectItf() function

```
RM2M_TXITF_NONE    = 0,          /* no uplink, communication with the server
                                    not possible */
RM2M_TXITF_MODEM   = 1,          // Mobile network modem
RM2M_TXITF_WIFI    = 2,          // WiFi module
RM2M_TXITF_LAN     = 3,          // LAN interface
```

Signal strength measurement flags

Control flags for the rM2M_GSMGetRSSI() and rM2M_GetRSSI() functions.

```
RM2M_RSSI_EXTENDED_VALUE = 0b00000001,  /* activates the extended value range
                                           (-32768 .. 32767) for the return
                                           values of the signal strength */
```

Configuration flags for the rM2M_CfgInit() function

```
RM2M_CFG_VOLATILE  = 0b00000001, // volatile storage (RAM)
```

15.2.3.3 Callback functions

public func(const data[], len, timestamp, timestamp256);

Function to be provided by the device logic developer, that is called up, once a data record has been read (using the function "rM2M_ReadData()") from the internal flash memory.

> ***Important note:*** *The parameter "timestamp256" has only been added in later firmware versions. The number of arguments transferred from the firmware to the callback function should thus be checked via the function "numargs()".*
>
> *Example:*

```
#callback readdata_callback(const data{}, len, timestamp, timestamp256)
{
  if(numargs() >= 4)
  {
    // parameter timestamp256 is available ...
  }
}
```

Parameter	*Explanation*
data	*Array that contains the data of the read data record*
len	*Length of the data area of the read data record in bytes (max. 1024 Byte)*
timestamp	*Time stamp of the data record (in UTC)*
timestamp256	*Fraction of the next started sec. (Resolution 1/256 sec.)*

public func(cfg);

Function to be provided by the script developer, that is called up if one of the configuration memory blocks has changed.

Parameter	*Explanation*
cfg	*Number of the changed configuration memory block starting with 0 for the first memory block*

15.2.3.4 Functions

native rM2M_TxStart(flags=0);

triggers a connection to the server with subsequent synchronisation of all memory areas (measurement data, configuration, position data, device log, files,...) between the device and the server. Only those memory areas are transmitted whose content has been changed. If the device is in "online" mode and an active connection to the server is established then this function only triggers synchronisation. The established connection is not disconnected beforehand and then re-established.

> ***Important note:*** *In "online" mode new measurement data that are stored in the internal flash via the "rM2M_RecData()" function are transferred to the server immediately. Calling the "rM2M_TxStart()" function is thus not necessary to transfer the measurement data in this case. Calling the function and the related synchronisation of all memory areas after generating every single measurement data record would lead to a much higher volume of data. The same also applies to transfer of the configurations. However it is recommended to call the "rM2M_ TxStart()" function occasionally (e.g. every 2h) even in "online" mode since not all memory areas are automatically synchronised.*

Parameter	***Explanation***	
flags	*Configuration flags for the connection establishment*	
	Bit0 (RM2M_TX_POSUPDATE):	*If set, the GSM position data is also updated.*
	Bit2 (RM2M_TX_REFRESH_ CONFIG):	*If set, a connection to the maintenance server is also established*
	Bit3 (RM2M_TX_SUPPRESS_ POSUPDATE):	*If set, this suppresses the update of the GSM position data that is automatically executed by the firmware every 24h*
	Bit4 (RM2M_TX_POSUPDATE_ ONLY):	*If set, only the GSM position data is updated when a connection is established. Measurement data, configurations etc. are not synchronised.*

	Explanation
Return value	• *OK, if successful* • *ERROR_SIM_STATE, if a connection is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 139)* • *ERROR_MODEM_DISABLED, if the connection cannot be established due to the supply voltage being too low* • *ERROR_TXITF, if the connection cannot be established due to the TX interface configuration (e.g. TX interface not open)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_TxSetMode(mode, flags=0);

Sets the connection type to be used. If the connection type is changed to "Online" or "Interval & wakeup", this is immediately followed by a connection being established incl. a synchronisation with the server, as long as this is not suppressed by the "RM2M_TXMODE_SUPPRESS_SYNC" flags being set. The same also applies to changing the connection type from "Interval" to "Interval & wakeup".

Parameter	***Explanation***
mode	*Connection type to be used:* *RM2M_TXMODE_TRIG: The connection is established when the "rM2M_TxStart()" function is called* *RM2M_TXMODE_WAKEUP: The connection is established in the same way as in "Interval" mode when the "rM2M_TxStart()" function is called. Additionally, the device can be initiated via the server to immediately establish a connection (see "myDatanet Server Manual " 805002). For this purpose, the device immediately logs into the GSM network as soon as this mode has been set.*  *RM2M_TXMODE_ONLINE: The device does not disconnect the connection and continuously transmits the measurement data. However, every 7 days, the connection is temporarily interrupted in order to verify the server assignment. The connection is established as soon as this mode has been set. Calling the "rM2M_TxStart()" function is not necessary.*
flags	*Configuration flags for the communication mode* *Bit0: automatic sync. with the server when the connection type is changed* *0 = Execute synchronisation* *RM2M_TXMODE_SUPPRESS_SYNC = Suppress synchronisation*

	Explanation
Return value	• *OK, if successful* • *ERROR_SIM_STATE, if the mode is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 139)* • *ERROR_MODEM_DISABLED, if the connection cannot be established due to the supply voltage being too low* • *ERROR_TXITF, if the connection cannot be established due to the TX interface configuration (e.g. TX interface not open)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

Note: *Additional explanation about the connection types*

Connection type	*Energy consumption*	*Data volumes*	*Response time*
online			
Interval & wakeup			
Interval			

native rM2M_TxGetStatus(&errorcode=0);

Returns the current connection status

Parameter	*Explanation*	
errorcode	*Variable to store the error code that occurred during the last connection attempt*	
	RM2M_TXERR_NONE:	*Last connection establishment successful*
	> RM2M_TXERR_NONE:	*Last connection establishment failed. Detailed breakdown of the error codes is provided in "Connection error codes" in chapter "Uplink" on page 138.*

	Explanation	
Return value	*Bit0 (RM2M_TX_FAILED):*	*set if the last GPRS connection establishment failed*
	Bit1 (RM2M_TX_ACTIVE):	*set when a GPRS connection is established*
	Bit2 (RM2M_TX_STARTED):	*set when a connection establishment has been started*
	Bit3 (RM2M_TX_RETRY):	*set during the delay until the next automatic retry in the event of connection problems*
	Bit4 (RM2M_TX_WAKEUPABLE):	*set when the modem is logged into the GSM network (wakeup possible)*
	Bit5 (RM2M_TX_EXTSIM):	*set if the external SIM is used for the connections*
	Bit6 (RM2M_TX_DISABLED):	*set if the modem has been deactivated*
	Bit8 (RM2M_TX_WAKEUP):	*Set when a connection establishment was triggered upon receipt of a wakeup SMS*
	Bit9 (RM2M_TX_POSUPDATE_ ACTIVE):	*set when positioning is running*

native rM2M_TxSelectItf(itf);

Selects the communication interface to be used for the uplink

Parameter	*Explanation*
itf	*Selection of the communication interface* *RM2M_TXITF_NONE:* *No uplink, communication with the server not possible* *RM2M_TXITF_MODEM:* *Mobile network modem* *RM2M_TXITF_WIFI:* *WiFi module* *RM2M_TXITF_LAN:* *LAN interface*

	Explanation
Return value	• *OK, if successful* • *ERROR, if the selected communication interface is not supported by the device or another error occurs*

native rM2M_TxItfGetStats(stats[TrM2M_TxItfStats], len=sizeof stats);

Returns the statistical information on the uplink communication interface

Parameter	*Explanation*
stats	*Structure for storing the statistical information (see "TrM2M_TxItfStats" in chapter "Arrays with symbolic indices" on page 138)*
len	*Size (in cells) of the structure to store the statistical information – OPTIONAL*

	Explanation
Return value	• *OK, if successful*

native rM2M_SetTCPKeepAlive(time=0);

Sets the interval at which the keep alive pings are sent during online mode

Parameter	*Explanation*
time	*Sets the time between the keep alive pings* *0:* *Default setting saved in the firmware is used (15 min. 3 sec.)* *< 241:* *in 1 sec. increments* *241 .. 255:* *in 5 min. increments, whereby 3 sec. is subsequently added* *e.g. 243: 3*5 min. + 3 sec. = 15 min. 3 sec.*

	Explanation
Return value	• *OK, if successful*

native rM2M_GSMGetRSSI(flags=0);

Returns the GSM/UMTS/LTE signal strength

> ***Important note:*** *Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects. The "rM2M_GetRSSI()" function should be used as an alternative.*

Parameter	***Explanation***
flags	*Configuration flags for the signal strength measurement* *Bit0 (RM2M_RSSI_EXTENDED_VALUE):* *If set, the extended value range for the return of the signal strength is used*

	Explanation
Return value	*Signal strength in [dBm]* *RM2M_RSSI_EXTENDED_VALUE not set:* • *Maximum value range: -127 to 127* • *Out of range at: -128* *RM2M_RSSI_EXTENDED_VALUE set:* • *Maximum value range: -32767 to 32767* • *Out of range at: -32768* *GSM values range from -113 to -51 dBm.* *UMTS values range from -116 to -54 dBm.* *LTE values range from -141 to -44 dBm.*

native rM2M_GetRSSI(flags=0);

Returns the signal strength at the communication interface used for the uplink

	Explanation
Return value	*Signal strength in [dBm]* *RM2M_RSSI_EXTENDED_VALUE not set:* • *Maximum value range: -127 to 127* • *Out of range at: -128* *RM2M_RSSI_EXTENDED_VALUE set:* • *Maximum value range: -32767 to 32767* • *Out of range at: -32768* *GSM values range from -113 to -51 dBm.* *UMTS values range from -116 to -54 dBm.* *LTE values range from -141 to -44 dBm.* *When using the LAN interface the return value is 0 dBm.*

native rM2M_GSMGetInfo(info[TrM2M_GSMInfo], len=sizeof info);

Returns information on the GSM modem, SIM chip and the GSM network used during the last connection

Parameter	*Explanation*
info	*Structure for storing the information (see "TrM2M_GSMInfo" in chapter "Arrays with symbolic indices" on page 138)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation
Return value	• *Used size (in cells) of the structure for storing the information* • *ERROR if the address and/or length of the info structure are invalid (outside the script data memory)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_LiveData(const data{}, len);

Transmits a data record as live data to the server. Calling this function is only permissible if the device is in "online" mode and an active connection to the server is established. Use the "rM2M_Pack", "rM2M_SetPacked" or "rM2M_SetPackedB" functions to generate the data area.

Parameter	*Explanation*
data	*Array that contains the live data to be transferred* ***Important note:*** *The structure of the live data must be identical to that of the measurement data saved in the internal flash using the "rM2M_RecData()" function.*
len	*Number of bytes to be transferred (max. 1024 Byte)*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs (e.g. the server does not support receipt of live data.)*

native rM2M_RecData(timestamp, const data{}, len);

Saves a data record in the internal flash memory. Use the "rM2M_Pack", "rM2M_SetPacked" or "rM2M_SetPackedB" functions to generate the data area.

Parameter	*Explanation*
timestamp	*Time stamp that should be used for the recording* *= 0: The current system time is used as the time stamp.* *> 0: The transferred time stamp is used.* *(The time stamp must be specified in seconds since 31.12.1999)*
data	*Array that contains the data to be saved*
len	*Number of bytes to be saved (max. 1024 Byte)*

	Explanation

Return value	• *OK, if successful*
	• *-2, if data storage is not currently possible as the internal memory is being reorganised. The data must be temporarily saved in the script and stored again at a later date.*
	• *ERROR, if one of the following errors occurs*
	  • *Memory area (data{}, len) is invalid.*
	  • *More than 10 calls during one script run.*
	  • *Number of bytes to be saved > 1024 Byte*
	  • *FLASH write process not successful*
	  • *The transfer parameter "timestamp" is more than 5 minutes in the future*
	• *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_ReadData(recidx, funcidx);

Reads out a data record saved in the internal flash and then calls up the function for which the index was transferred.

Parameter	*Explanation*
recidx	*Index of the data record to be read (-1 = last/current data record, -2 = penultimate data record, ...)*
funcidx	*Index of the public function that should be called once the data record has been read from the internal flash memory.* *Type of function: public func(const data[], len, timestamp, timestamp256);*

	Explanation
Return value	• *OK, if the read process has been started* • *ERROR, if an error occurs*

native rM2M_CfgInit(cfg, flags);

Sets the configuration for a configuration memory block. Calling the function is only necessary if one of the configuration flags should be set.

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*
flags	*Configuration flags to be set/deleted* *Bit0: Type of storage* *0 (default) = stored in FLASH in non-volatile manner* *RM2M_CFG_VOLATILE = saved in RAM in volatile manner*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

Note: *Additional explanation on the type of storage:*

If Bit 0 was not set (default), the non-volatile storage of the configuration memory block in the FLASH is initiated when the "rM2M_CfgWrite" function is called up.

If Bit0 was set (Bit0 = RM2M_CFG_VOLATILE), the configuration memory block is saved in the RAM in a volatile manner when the "rM2M_CfgWrite" function is called. This option is recommended if the data in the configuration memory block changes frequently as this will reduce the number of flash write cycles. The "rM2M_CfgFlush" function must be called up so that the configuration memory block is saved in a non-volatile manner in the FLASH.

native rM2M_CfgWrite(cfg, pos, const data{}, size);

Saves the transferred data block at the specified position in a configuration memory block. Note that the configuration memory block is either saved in the RAM in a volatile manner (Bit0 = RM2M_CFG_ VOLATILE) or in the FLASH in a non-volatile manner (Bit0 = 0, default) depending on the type of storage selected via the "rM2M_CfgInit" function. The function is also passed which of the 10 available memory blocks in the internal flash memory should be used. Use the "rM2M_Pack", "rM2M_ SetPacked" or "rM2M_SetPackedB" functions to generate the data block that should be saved. The time stamp is updated, so that the configuration memory block is automatically synchronised with the myDatanet server during the next connection.

Parameter	***Explanation***
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*
pos	*Byte offset within the configuration memory block to determine the position where the data should be written.*
data	*Array that contains the data that should be written in the configuration memory block*
size	*Number of bytes that should be written in the configuration memory block*

	Explanation
Return value	• *> 0: Current size of the configuration memory block if successful* • *ERROR_MEM, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_CfgFlush(cfg);

Saves the configuration memory block for which the number was transferred in the FLASH in a non-volatile manner. Calling the function is only necessary if "volatile in RAM (Bit0 = RM2M_CFG_ VOLATILE)" was selected as the type of storage for the relevant configuration memory block via the "rM2M_CfgInit" function.

Parameter	***Explanation***
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native rM2M_CfgRead(cfg, pos, data{}, size);

Reads a data block from the specified position in a configuration memory block. The function is also informed which of the 10 available memory blocks in the internal flash memory should be read. Use the "rM2M_Pack", "rM2M_GetPacked" or "rM2M_GetPackedB" functions to unpack the read data.

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*
pos	*Byte offset within the configuration memory block to determine the position from which the data should be read*
data	*Array to store the data to be read*
size	*Number of bytes that should be read from the configuration memory block*

	Explanation
Return value	• *>0: Number of bytes actually read. This may be less or equal to the requested number of bytes.* • *ERROR_MEM, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_CfgDelete(cfg);

Deletes all of the data of the transferred configuration memory block

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*

	Explanation
Return value	• *OK, if successful* • *ERROR_MEM, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

native rM2M_CfgOnChg(funcidx);

Specifies the function that should be called if one of the configuration memory blocks has changed

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up if the configuration has changed* *Type of function: public func(cfg);*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index of a public function was transferred* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

15.2.4 System

15.2.4.1 Arrays with symbolic indices

TEasyV3_SysValue

Internal measurement values

```
// Temp       Internal housing temperature in [0.1°C]
// RH         Humidity in the housing in [0.1% rH]

#define TEasyV3_SysValue[.Temp, .RH]
```

15.2.4.2 Functions

native EasyV3_GetSysValues(values[TEasyV3_SysValue], len=sizeof values);

Reads the last valid values for the "Internal housing temperature" and "Air humidity in the housing" from the system. The interval for determining these values is 10sec. and cannot be changed.

Parameter	*Explanation*
values	*Structure for storing the measurement values (see "TEasyV3_SysValue" in chapter "Arrays with symbolic indices" on page 154)*
len	*Size (in cells) of the structure to store the measurement values - OPTIONAL*

	Explanation

Return value	• *Used size (in cells) of the structure for storing the measurement values*
	• *ERROR, if one of the following errors occurs*
	  ◦ *Address and/or length of the values structure is invalid (outside the script data memory).*
	  ◦ *One of the measurement values is invalid.*

15.2.5 Encoding

15.2.5.1 Constants

Configuration flags for the rM2M_Pack() function

```
RM2M_PACK_GET      = 0b00000001, // Value should be read (get packed)
RM2M_PACK_BE       = 0b00000010, // Use "Big endian" format
RM2M_PACK_U8       = 0b00010000, // 8-bit unsigned
RM2M_PACK_S8       = 0b10010000, // 8-bit signed
RM2M_PACK_U16      = 0b00100000, // 16-bit unsigned
RM2M_PACK_S16      = 0b10100000, // 16-bit signed
RM2M_PACK_U32      = 0b01000000, // 32-bit unsigned
RM2M_PACK_S32      = 0b11000000, // 32-bit signed
RM2M_PACK_F32      = 0b01000000, // 32-bit float
```

15.2.5.2 Functions

native rM2M_SetPacked(data{}, pos, &{Float,Fixed,_}:value, size=4, bool:bigendian=false);

Writes the transferred value to a specified position in an array

> ***Important note:*** *Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects as the signed data types might lead to problems. The „rM2M_Pack()" function should be used as an alternative.*

Parameter	***Explanation***
data	*Array that should be used as a data area for a data record or a configuration*
pos	*Byte offset within the array to determine the position where the value should be written*
value	*Value that should be written in the array*
size	*Number of bytes that should be used for the value to be written*
bigendian	*Settings for the byte sequence that should be used when writing the value:* *true: "Big endian" is used* *false: "Little endian" is used*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

Note: *Additional explanation on the byte sequence:*

In the following example, the whole number 439.041.101 is saved as a 32-bit integer value from memory address 10000.

	Big endian			***Little endian***		
Addresses	***Hex***	***Dez***	***Binary***	***Hex***	***Dez***	***Binary***
10000	*1A*	*26*	*00011010*	*4D*	*77*	*01001101*
10001	*2B*	*43*	*00101011*	*3C*	*60*	*00111100*
10002	*3C*	*60*	*00111100*	*2B*	*43*	*00101011*
10003	*4D*	*77*	*01001101*	*1A*	*26*	*00011010*

native rM2M_SetPackedB(data{}, pos, const block{}, size);

Writes the transferred data block to the specified position in an array

Parameter	*Explanation*
data	*Array that should be used as a data area for a data record or a configuration*
pos	*Byte offset within the array to determine the position where the data block should be written*
block	*Data block that should be written in the array*
size	*Number of bytes to be written from the data block to the array*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native rM2M_GetPacked(const data{}, pos, &{Float,Fixed,_}:value, size=4, bool:bigendian=false);

Supplies the value that is located at the specified position in an array

Important note: *Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects as the signed data types might lead to problems. The „rM2M_Pack()" function should be used as an alternative.*

Parameter	***Explanation***
data	*Array that should be used as a data area for a data record or a configuration*
pos	*Byte offset within the array to determine the position from which the data should be read*
value	*Variable to store the data to be read*
size	*Number of bytes that should be read*
bigendian	*Specifies how the packed data must be interpreted:* *true: The data is saved in "Big endian" format in the array.* *false: The data is saved in "Little endian" format in the array.*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

Note: *Additional explanation on the byte sequence:*

In the following example, the whole number 439.041.101 is saved as a 32-bit integer value from memory address 10000.

	Big endian			***Little endian***		
Addresses	***Hex***	***Dez***	***Binary***	***Hex***	***Dez***	***Binary***
10000	*1A*	*26*	*00011010*	*4D*	*77*	*01001101*
10001	*2B*	*43*	*00101011*	*3C*	*60*	*00111100*
10002	*3C*	*60*	*00111100*	*2B*	*43*	*00101011*
10003	*4D*	*77*	*01001101*	*1A*	*26*	*00011010*

native rM2M_GetPackedB(const data{}, pos, block{}, size);

Reads a data block that is located at the specified position in an array

Parameter	*Explanation*
data	*Array that should be used as a data area for a data record or a configuration*
pos	*Byte offset within the array to determine the position from which the data should be read*
block	*Array to store the data to be read*
size	*Number of bytes that should be read*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native rM2M_Pack(const data{}, pos, &{Float,Fixed,_}:value, type);

Function to access packed data. If the Bit0 (RM2M_PACK_GET) of the "type" parameter was set, the function returns the value that is located at the specified position in the array. Otherwise the function writes the transferred value to the specified position in the array.

Parameter	***Explanation***
data	*Array with the packed content* *Set packed: Array to which the value should be written* *Get packed: Array from which the value should be read*
pos	*Byte offset within the array* *Set packed: Position to which the value should be written* *Get packed: Position from which the value should be read*
value	*Set packed: Value that should be written in the array* *Get packed: Variable to store the data to be read*
type	*Configuration flags for the function* *Bit0: Select "Set packed" / "Get packed"* *0 = value should be written* *1 = value should be read* *Bit1: Byte order* *0 = "Little endian" format* *1 = "Big endian" format* *Bit2...3* *reserved for extensions* *Bit4...7: Data type* *1 = 8-bit unsigned* *2 = 16-bit unsigned* *4 = 32-bit unsigned / 32-bit float* *9 = 8-bit signed* *10 = 16-bit signed* *12 = 32-bit signed* ***Note:*** *You can also use the predefined constants for this parameter (see "Configuration flags for the rM2M_Pack() function" in chapter "Constants" on page 155). The constants can also be combined using the "or" link.*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

15.2.6 Bluetooth Low Energy

15.2.6.1 Arrays with symbolic indices

TBLE_Id

Information to identify the BLE module installed in the device. In contrast to the information that is contained in the "TBLE_DevInfo" structure, this information can be called up at any time via the "BLE_GetId()" function.

```
// FWVersion  Version of the application installed on the BLE module
//            The minor version is saved in byte 0, the major version is saved
//            in byte 1 z.B. "01v003": Major version = 1, Minor version = 3

#define TBLE_Id[.FWVersion]
```

TBLE_Scan

Information regarding a BLE peripheral found during a (passive or active) scan process

```
// addr_type  Type of BLE address
//            0: Public address
//            1: Random static address
//            2: Random private resolvable address
//            3: Random private non-resolvable address
// addr       HW address
// rssi       Signal strength [dBm]
// name       Advertising name of the peripheral
// msd_len    Number of bytes saved in "msd"
// msd        Manufacturer-specific data

#define TBLE_Scan[.addr_type, .addr{6}, .rssi, .name{32+1}, .msd_len, .msd{32}]
```

TBLE_ScanFinished

Information on how a scan process (passive or active) was completed

```
// result  specifies how the scan process was terminated
//         OK:   Scan process completed successfully
//         ERROR: An error occurred
//         ERROR-1: Timeout

#define TBLE_ScanFinished[.result]
```

TBLE_Notify

Content/data of a characteristic for which the notification was activated

```
// charHandle  Handle of a characteristic
// len         Number of received bytes
// data        Array that contains the received data

#define TBLE_Notify[.charHandle, .len, .data{BLE_MAXLEN_NOTIFY}]
```

TBLE_WriteResult

Information on how the write process on a characteristic was completed

```
// result      specifies how the write process was terminated
//             OK:    Write process completed successfully
//             ERROR: An error or timeout occurred
// charHandle  Handle of the characteristic
// len         Number of written bytes, if successful

#define TBLE_WriteResult[.result, .charHandle, .len]
```

TBLE_Read

Block of requested data from a characteristic

```
// charHandle  Handle of a characteristic
// offset      Offset of the received data block within the characteristic
// len         Number of available bytes in the "data" array
// data        Array that contains the received data

#define TBLE_Read[.charHandle, .offset, .len, .data{BLE_MAXLEN_READ}]
```

TBLE_ReadResult

Information on how the read process on a characteristic was completed

```
// result      specifies how the read process was terminated
//             OK:    Read process completed successfully
//             ERROR: An error or timeout occurred
// charHandle  Handle of the characteristic

#define TBLE_ReadResult[.result, .charHandle]
```

TBLE_DevInfo

Information to identify the BLE module installed in the device. This information is provided once via the callback function ("BLE_EVENT_DEVINFO" event) if the BLE module is initialised via BLE_Init().

```
// chipname    Ascii, designation of the BLE module (e.g. NRF52)
// appname     Ascii, designation of the application installed on the BLE module
//             (e.g. ScannerApp)
// hwrev       Ascii, HW revision (e.g. "01v001") (reserved for extensions)
// fwrev       Ascii, FW version (z.B. "02v000") of the application installed on
//             the BLE module
// addr        HW address (all bytes are zero, if not available)

#define TBLE_DevInfo[.chipname{BLE_MAXLEN_CHIPNAME+1},
                     .appname{BLE_MAXLEN_APPNAME+1}, .hwrev{BLE_MAXLEN_HWREV+1},
                      .fwrev{BLE_MAXLEN_FWREV+1}, .addr{6}]
```

TBLE_Connect

Information regarding the status change of the connection with a BLE peripheral

```
// result      Status of the connection
//             1: disconnected
//             0: connected
//            <0: ERROR
// addr        HW address of the BLE peripheral

#define TBLE_Connect[.result, .addr{6}]
```

TBLE_Error

Information regarding which error has occurred

```
// errorcode  error code (see "BLE error codes" in chapter
//            "Constants" on page 163)

#define TBLE_Error[.errorcode]
```

15.2.6.2 Constants

Maximum BLE connections

Maximum number of simultaneous BLE connections

```
BLE_MAX_CONNECTIONS = 5
```

Status of the BLE interface

Return values of the BLE_GetState() function

```
BLE_STATE_OFF   = 0,  // The BLE interface is switched off
BLE_STATE_INIT  = 1,  // Init sequence active, not ready (yet)
BLE_STATE_READY = 2,  // Ready for the receipt of commands
BLE_STATE_BUSY  = 3   // A command is currently being processed
```

BLE events

Possible BLE events for function type "public func(event, connhandle, const data{}, len);" to be provided by the script developer

```
BLE_EVENT_SCAN          = 0, /* BLE Central: Scan information available (passive
                                scan) */
BLE_EVENT_SCAN_RSP      = 1, /* BLE Central: Scan information available (active
                                scan) */
BLE_EVENT_NOTIFY        = 2, // BLE Central: Notification received
BLE_EVENT_READ          = 3, // BLE Central: Read response data available
BLE_EVENT_WRITE_RESULT  = 4, // BLE Central: Write command is terminated
BLE_EVENT_READ_RESULT   = 5, // BLE Central: Read command is terminated
BLE_EVENT_SCAN_FINISHED = 6, // BLE Central: The scan process is terminated
BLE_EVENT_DEVINFO       = 7, /* Device information available (BLE module is
                                ready) */
BLE_EVENT_CONNECT       = 8, /* BLE Central: Connected to/disconnected from
                                peripheral */
BLE_EVENT_ERROR         = 9, // Internal error has occurred
BLE_EVENT_RAWCMD_STRING = 10,// RAW command: STRING/ASCII response available
BLE_EVENT_RAWCMD_OK     = 11,/* RAW command: OK response available (command
                                terminated successfully) */
BLE_EVENT_RAWCMD_ERROR  = 12,/* RAW command: ERROR response available (command
                                failed) */
```

Size of the notification data area

Maximum size (in bytes) of the data area of a notification

```
BLE_MAXLEN_NOTIFY = 256
```

Size of the data area of a read data block

Maximum size (in bytes) of the data area of a requested data block from a characteristic

```
BLE_MAXLEN_READ = 256
```

String lengths for the TBLE_DevInfo structure

```
#define BLE_MAXLEN_CHIPNAME      (20)          /* Max. length of the designation
                                                  for the BLE module */
#define BLE_MAXLEN_APPNAME       (20)          /* Max. length of the designation
                                                  for the application installed
                                                  on the BLE module */
#define BLE_MAXLEN_HWREV         (20)          /* Max. length for the string */
                                                  that specifies the HW revision
#define BLE_MAXLEN_FWREV         (20)          /* Max. length for the string
                                                  that specifies the FW
                                                  revision */

#define BLE_CHIPNAME_NRF52       "NRF52"       /* Designation of the BLE module
                                                  installed in the device */
#define BLE_APPNAME_SCANNER_APP "ScannerApp"   /* Designation of the application
                                                  installed on the BLE module */
```

BLE error codes

```
BLE_ERROR_UNKNOWN      = 0, // Unspecified internal error
BLE_ERROR_INIT         = 1, // Initialisation error
BLE_ERROR_IO           = 2, // Internal IO error
BLE_ERROR_RESTART      = 3, // Automatic restart detected/forced
BLE_ERROR_FORCED_DISC  = 4, // Forced disconnect error
```

15.2.6.3 Callback functions

public func(event, connhandle, const data{}, len);

Function to be provided by the script developer, that is called up if a BLE event has occurred

Parameter	*Explanation*
iEvent	*BLE event that has occurred (see "BLE events" in chapter "Constants" on page 163"*
connhandle	*Connection handle of a certain BLE peripheral*
data *(1/2)*	*Array for which the content is dependent on the type of BLE event:* • *BLE_EVENT_SCAN: Information on a BLE peripheral found during a scan process (see "TBLE_Scan" structure)* *This event is triggered if a BLE peripheral was found during scanning (passive) (see BLE_Scan()).* • *BLE_EVENT_SCAN_RSP: Information on a BLE peripheral found during a scan process (see "TBLE_Scan" structure)* *This event is triggered if a BLE peripheral was found during scanning (active) (see BLE_Scan()).* • *BLE_EVENT_NOTIFY: Content/data of a characteristic for which the notification was activated (see "TBLE_Notify" structure)* *This event is triggered if the software of the BLE peripheral changes the content of a characteristic for which the notification was previously activated.* • *BLE_EVENT_READ: Block of required data from a characteristic (see "TBLE_Read" structure)* *This event is triggered if a block of the requested data was received (see BLE_Read()).* • *BLE_EVENT_WRITE_RESULT: Information on how the write process on a characteristic was completed (see "TBLE_WriteResult" structure).* *This event is triggered if the data was successfully written in the characteristics of the BLE peripheral or an error occurred during this process (see BLE_Write()).* • *BLE_EVENT_READ_RESULT: Information on how the read process on a characteristic was completed (see "TBLE_ReadResult" structure).* *This event is triggered if all of the requested data was received or an error occurred during this process (see BLE_Read()).* • *BLE_EVENT_SCAN_FINISHED:" Information on how a scan process was completed (see "TBLE_ScanFinished" structure)* *This event is triggered when the scan process is completed (see BLE_ Scan()).*

Parameter	Explanation
data *(2/2)*	• *BLE_EVENT_DEVINFO: Information to identify the BLE module installed in the device (see "TBLE_DevInfo" structure)* *This event is triggered if the BLE module has been initialised via BLE_Init() and is ready.* • *BLE_EVENT_CONNECT: Information regarding the status change of the connection with a BLE peripheral (see "TBLE_Connect" structure)* *This event is triggered if the connection to a BLE peripheral could be established/disconnected or if an error was triggered (see BLE_Connect() and BLE_Disconnect()).* • *BLE_EVENT_ERROR: Information regarding which error has occurred (see "TBLE_Error" structure)* *This event is triggered if an internal error occurs.* • *BLE_EVENT_RAWCMD_STRING: String that was sent by the BLE module as a response to the RAW command.* *This event is triggered if a STRING/ASCII response is available for a RAW command sent via BLE_SendRawCmd() to the BLE module* • *BLE_EVENT_RAWCMD_OK: "OK"* *This event is triggered if a RAW command sent via BLE_SendRawCmd() to the BLE module has been terminated successfully.* • *BLE_EVENT_RAWCMD_ERROR: "ERROR"* *This event is triggered if a RAW command sent via BLE_SendRawCmd() to the BLE module could not be processed or failed.* ***Note:** The descriptions of the structures mentioned above are detailed in the chapter "Arrays with symbolic indices" on page 161, while the functions are detailed in chapter "Functions" on page 168*
iDataLen	*Number of bytes in the array*

15.2.6.4 Functions

native BLE_GetId(id[TBLE_Id], len=sizeof id);

Returns the information to identify the BLE module installed in the device

Parameter	*Explanation*
id	*Structure for storing the information to identify the BLE module installed in the device (see "TBLE_Id" in chapter "Arrays with symbolic indices" on page 161)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation

Return value	• *Used size (in cells) of the structure for storing the information*
	• *ERROR, if one of the following errors occurs*
	  • *if the address and/or length of the ID structure are invalid (outside the script data memory)*
	  • *the BLE module is not ready*

native BLE_Init(funcidx);

initialises the BLE interface and specifies the function that should be called up if a BLE event has occurred

Note: *The "BLE_EVENT_DEVINFO" event is triggered as soon as the BLE module is ready.*

Parameter	*Explanation*
funcidx	*Index of the public function that is called up if a BLE event has occurred* *Type of function: public func(event, connhandle, const data{}, len);*

	Explanation
Return value	• *OK, if successful* • *ERROR_FEATURE_LOCKED, if the BLE module is not unlocked on the device* • *ERROR, if an error occurs*

native BLE_Close();

Deactivates the BLE interface. In doing so, the connection to the currently connected sensors is also automatically disconnected.

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native BLE_GetState();

Returns the current status of the BLE interface

	Explanation
Return value	*Status of the BLE interface (see "Status of the BLE interface" in the chapter "Constants" on page 163)*

native BLE_Scan(time = 10, flags = 0);

starts the scan process according to the BLE peripherals

> ***Note:*** *The command works in non-blocking mode. The "BLE_EVENT_SCAN" event (passive scan) or "BLE_EVENT_SCAN_RSP" event (active scan) is triggered every time a BLE peripheral has been found. The "BLE_EVENT_SCAN_FINISHED" event is triggered when the scan process is completed (i.e. the scan duration has expired). Once the scan process is completed, the BLE_GetState() function returns "BLE_STATE_READY" instead of "BLE_STATE_BUSY".*

Parameter	***Explanation***
time	*Scan duration [s]*
flags	*Type of BLE scan* • *0 passive scan (most energy-efficient)* • *1 active scan (scan request)*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_Connect(addr{6}, itv = -1);

Establishes the connection to a BLE peripheral

Note: *The command works in non-blocking mode. The "BLE_EVENT_CONNECT" event is triggered if*

- *the connection to a BLE peripheral could be established,*
- *an existing connection was interrupted or*
- *an error occurred when establishing a connection.*

Parameter	***Explanation***
addr	*HW address of the BLE peripheral with which the connection should be established.*
itv	*BLE connection interval in [ms] (i.e. interval at which the data is exchanged)* *Valid values: 8...1000 ms*

	Explanation
Return value	• *>= 0: Connection handle of a certain BLE peripheral* • *ERROR-1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_Disconnect(connhandle = 0);

Disconnects the connection to a BLE peripheral

Note: *The BLE peripheral must already be connected*

Note: *The command works in non-blocking mode. The "BLE_EVENT_CONNECT" event is triggered if*

- *the connection to a BLE peripheral could be disconnected,*
- *an error occurred when disconnecting the connection.*

Parameter	***Explanation***
connhandle	*Connection handle of a certain BLE peripheral*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_GetConnState(connhandle = 0);

Returns the connection status for a certain BLE peripheral

Parameter	*Explanation*
connhandle	*Connection handle of a certain BLE peripheral*

	Explanation

Return value	• *>0: Connected*
	• *0: Connection disconnected*
	• *ERROR-1, if the BLE interface has not been initialised via BLE_Init()*
	• *ERROR -2, if the command queue can no longer store any more commands*
	• *ERROR*
	  • *Invalid handle*
	  • *BLE module not active*

native BLE_Write(connhandle = 0, handle, const data{}, size);

Writes data in a certain characteristic of a BLE peripheral

Note: *The BLE peripheral must already be connected*

Note: *The command works in non-blocking mode. The "BLE_EVENT_WRITE_RESULT" event is triggered if the data was successfully written in the characteristic of the BLE peripheral or an error occurred while the data was being written.*

Parameter	*Explanation*
connhandle	*Connection handle of a certain BLE peripheral*
handle	*Handle of a certain characteristic (currently has to be determined externally)*
data	*Array that contains the data to be written in the characteristic*
size	*Number of bytes to be written*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_Read(connhandle = 0, handle);

Reads data from a certain characteristic of a BLE peripheral

Note: *The BLE peripheral must already be connected*

Note: *The command works in non-blocking mode. The "BLE_EVENT_READ" event is triggered every time a block of the requested data has been received. The "BLE_EVENT_READ_RESULT" event is triggered if all of the requested data was received or an error occurred while reading the data.*

Parameter	***Explanation***
connhandle	*Connection handle of a certain BLE peripheral*
handle	*Handle of a certain characteristic (currently has to be determined externally)*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_ChgConItv(connhandle = 0, conitv);

Changes the BLE connection interval (i.e. interval at which the data is exchanged)

Note: *The BLE peripheral must already be connected*

Parameter	***Explanation***
connhandle	*Connection handle of a certain BLE peripheral*
conitv	*BLE connection interval in [ms] (i.e. interval at which the data is exchanged)* *Valid values: 8...1000 ms*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_SetScanResponseData(const data{}, size);

is used to set the manufacturere-specific data which should be included in the response to an active scan carried out by a BLE Central.

Parameter	*Explanation*
data	*Array with the data which will be included as "manufacturer-specific data" in the response to an active scan carried out by a BLE Central*
size	*Size (in cells) of the transferred array*

	Explanation
Return value	• *OK, if successful* • *ERROR -1, if the BLE interface has not been initialised via BLE_Init()* • *ERROR-2, if the command queue can no longer store any more commands* • *ERROR, if another error occurs*

native BLE_SendRawCmd(cmd{}, cmdlen, timeout=0);

Sends RAW commands to the BLE module. A RAW command is forwarded directly to the BLE module by the firmware. The developer of the device logic is responsible for developing the RAW commands that are suitable for the BLE module (e.g. stop advertising on NRF5x scanner app: "at+advertise=0\r\n").

> ***Note:*** *The command works in non-blocking mode. Depending on whether the BLE module could process the RAW command, one of the following events is triggered:*
>
> - *BLE_EVENT_RAWCMD_STRING: STRING/ASCII response available*
> - *BLE_EVENT_RAWCMD_OK: Command completed successfully*
> - *BLE_EVENT_RAWCMD_ERROR: Command failed*

Parameter	*Explanation*
cmd	*Array that contains the RAW commands to be sent*
cmdlen	*Number of bytes that make up the RAW command* ***Note:*** *If the BLE module expects ASCII commands (e.g. NRF5x scanner app), then the final '\0' after the RAW command must be added to the RAW command by the user.*
timeout	*Response timeout* • *0: Internal default timeout is used* • *>0: Response timeout in [ms]*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native BLE_Setbuf(rxbuf{}, rxlen, txbuf{}, txlen);

Provides the firmware with one buffer for sending and one for receiving characters via the BLE interface from the RAM area reserved for the device logic. When this function is called up, the system switches from the 256 bytes buffers integrated in the firmware to the transferred buffers.

> ***Important note:*** *If required, this function must be called via the "BLE_Init()" function before the BLE interface is initialised.*
>
> ***Important note:*** *The buffers "rxbuf" and "txbuf" must be valid during the entire use by the firmware (i.e. they must be defined as a global or static variable).*

Parameter	***Explanation***
rxbuf	*Static byte array that should be used as a buffer to receive characters via the BLE interface*
rxlen	*Size of the receiving buffer in byte* ***Note:*** *If the function is called up again and the size is set to "0" during the process, then the system switches back to the integrated buffer (256 bytes). The transferred static byte array can then be used by the device logic again.*
txbuf	*Static byte array that should be used as a buffer to send characters via the BLE interface*
txlen	*Size of the sending buffer in byte* ***Note:*** *If the function is called up again and the size is set to "0" during the process, then the system switches back to the integrated buffer (256 bytes). The transferred static byte array can then be used by the device logic again.*

	Explanation
Return value	• *OK, if successful* • *ERROR_FEATURE_LOCKED, if the BLE module is not unlocked on the device* • *ERROR, if an error occurs* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

15.2.7 Registry

15.2.7.1 Constants

Indices of the registration memory blocks

that can be accessed via the "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()", "rM2M_RegDelValue()" and "rM2M_RegDelKey()" functions. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

```
//System-specific data
RM2M_REG_SYS_OTP   = 0, /* Written once as part of the production
                           process (readonly by device logic). */
RM2M_REG_SYS_FLASH = 1, /* Can be changed during operation (readonly by
                           device logic) */

//Application-specific data
RM2M_REG_APP_OTP   = 2, /* Recommendation: Write only once as part of
                           the production process (readable and writeable
                           by device logic) */
RM2M_REG_APP_FLASH = 3, /* Can be changed during operation (readable and
                           writeable by device logic) */

//Application-specific, volatile data
RM2M_REG_APP_STATE = 4, /* Can be changed during operation (readable and
                           writeable by device logic). Requires
                           "rM2M_RegInit()" */

//Number of registration memory blocks
RM2M_REG_NUM_REGS  = 5,
```

Error codes for the registration memory block access operations

```
RM2M_REG_ERROR_TOKENMEM  = -101,  // Not enough tokens were provided
RM2M_REG_ERROR_INVAL     = -102,  // Invalid character inside JSON string
RM2M_REG_ERROR_PART      = -103,  /* The string is not a full JSON packet, more
                                     bytes expected */

RM2M_REG_ERROR_NOMEM     = -200,  // memory allocation failed
RM2M_REG_ERROR_NUMTOKENS = -201,  /* not enough token available for this
                                     object/array size */
RM2M_REG_ERROR_PAIR      = -202,  // found invalid pair (string : value)
RM2M_REG_ERROR_NOTOKENS  = -203,  // not enough tokens free for appending
RM2M_REG_ERROR_NOTFOUND  = -204,  // specified pair not found
RM2M_REG_ERROR_TYPE      = -205,  // token type mismatch
RM2M_REG_ERROR_PARAM     = -206,  // invalid parameters
RM2M_REG_ERROR_SIZE      = -207,  // size exceeds maximum allowed
RM2M_REG_ERROR_INVALID   = -208,  // JSON structure invalid
RM2M_REG_ERROR_ISNULL    = -209,  // value is null
```

Configuration flags for the rM2M_RegInit() function

```
RM2M_REG_VOLATILE  = 0b00000001, // volatile storage (RAM)
```

15.2.7.2 Callback functions

public func(reg);

Function to be provided by the device logic developer, that is called up if the registration has changed

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175) that has been changed*

15.2.7.3 Functions

native rM2M_RegInit(reg, flags, data{}, len=sizeof data);

initialises one of the optional registration memory blocks stored in the RAM. Calling up the function is only necessary for the registration memory blocks listed in the explanation of the "reg" parameter. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	*Explanation*
reg	*Registration memory block index* *The following registration memory blocks require an initialisation:* • *RM2M_REG_APP_STATE: Application-specific, volatile data (e.g. current device status)*
flags	*Configuration flags to be set/deleted* *Bit0: Type of storage* *0 = invalid, currently not supported* *RM2M_REG_VOLATILE = saved in RAM in volatile manner*
data	*Array to store the registration memory block*
len	*Size (in cells) of the transferred array to store the registration memory block (max. 1kB) - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegGetString(reg, const name[], string[], len=sizeof string);

Reads a character string from a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	Explanation
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)* ***Note:** RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.*
name	*Name of the entry*
string	*Array to store the string to be read* *(Extended JSON string format, see "rM2M_RegSetString ()" for details)*
len	*Size (in cells) of the transferred array to store the string to be read - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *RM2M_REG_ERROR_NOTFOUND, if the specified entry does not exist* • *RM2M_REG_ERROR_ISNULL if the value of the specified entry is set to "null"* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegGetValue(reg, const name[], &{Float,Fixed,_}:value, tag=tagof value);

reads a value from a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	Explanation
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)* ***Note:** RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.*
name	*Name of the entry*
value	*Variable to store the value to be read*
tag	*The integer and floating-point conversion are differentiated by the "tag" of the variables. - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *RM2M_REG_ERROR_NOTFOUND, if the specified entry does not exist* • *RM2M_REG_ERROR_ISNULL if the value of the specified entry is set to "null"* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegSetString(reg, const name[], const string[]);

Writes a character string into a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

> ***Important note:*** *This function accepts even characters which are fobidden according to JSON standard, such as "\t" , "\n" ,... Due to this, Javascript's JSON.parse() will fail with error messages. Use JSON5 instead to decode such extended strings.*

Parameter	***Explanation***
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)* ***Note:*** *RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.*
name	*Name of the entry* *If an entry with this name already exists, the existing character string is replaced by the transferred character string. Otherwise a new entry is created.*
string	*Array that contains the string to be written*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegSetValue(reg, const name[], {Float,Fixed,_}:value, tag=tagof value);

Writes a value into a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	***Explanation***
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)* ***Note:*** *RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.*
name	*Name of the entry* *If an entry with this name already exists, the existing value is replaced by the transferred value. Otherwise a new entry is created.*
value	*Value to be written*
tag	*The integer and floating-point conversion are differentiated by the "tag" of the value. - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegDelValue(reg, const name[]);

Searches for an entry based on its name and sets the value of this entry (regardless of whether it is a string or value) to "null". Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	***Explanation***
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)*
name	*Name of the entry for which the value should be set to "null"*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegDelKey(reg, const name[]);

Searches for an entry based on its name and deletes the entry from the registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	***Explanation***
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 175)*
name	*Name of the entry that should be deleted from the registration memory block*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an unspecified errors occurs* • *< OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 175)*

native rM2M_RegOnChg(funcidx);

Specifies the function that should be called up if one of the registration memory blocks has changed (i.e. has been updated by the server). The callback is not triggered upon local (device-side) changes of a registration memory.Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 45.

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up if the registration has changed* *Type of function: public func(reg);*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

15.2.8 Position

15.2.8.1 Arrays with symbolic indices

TrM2M_GSMPos

Information about a GSM/UMTS/LTE cell in the receiving range

```
// mcc        MCC (Mobile Country Code) of the GSM cell
// mnc        MNC (Mobile Network Code) of the GSM cell
// lac        LAC (Location Area Code) of the GSM cell
// cellid     Cell ID of the GSM cell
// rssi       Detected GSM level [dBm] for the GSM cell
// ta         TA (Timing Advance) of the GSM cell (currently always 0)

#define TrM2M_GSMPos[.mcc, .mnc, .lac, .cellid, .rssi, .ta]
```

TrM2M_PosUpdateGSM

Information about a GSM cell in the receiving range

```
// type       specifies the type of the entry (RM2M_POSUPDATE_TYPE_GSM)
// stamp      Time when data was recorded
// mcc        MCC (Mobile Country Code) of the GSM cell
// mnc        MNC (Mobile Network Code) of the GSM cell
// lac        LAC (Location Area Code) of the GSM cell
// cid        Cell ID of the GSM cell
// rssi       Detected GSM level [dBm] for the GSM cell
// ta         TA (Timing Advance) of the GSM cell (currently always 0)

#define TrM2M_PosUpdateGSM [.type, .stamp, .mcc, .mnc, .lac, .cid, .rssi, .ta]
```

TrM2M_PosUpdateUMTS

Information about a UMTS cell in the receiving range

```
// type       specifies the type of the entry (RM2M_POSUPDATE_TYPE_UMTS)
// stamp      Time when data was recorded
// mcc        MCC (Mobile Country Code) of the GSM cell
// mnc        MNC (Mobile Network Code) of the GSM cell
// lac        LAC (Location Area Code) of the GSM cell
// cid        Cell ID of the GSM cell
// rscp       Received Signal Code Power [dBm]
// pscr       Primary Scrambling Code

#define TrM2M_PosUpdateUMTS [.type, .stamp, .mcc, .mnc, .lac, .cid, .rscp,
                             .pscr]
```

TrM2M_PosUpdateLTE

Information about an LTE cell in the receiving range

```
// type       specifies the type of the entry (RM2M_POSUPDATE_TYPE_LTE)
// stamp      Time when data was recorded
// mcc        MCC (Mobile Country Code) of the GSM cell
// mnc        MNC (Mobile Network Code) of the GSM cell
// lac        LAC (Location Area Code) of the GSM cell
// cid        Cell ID of the GSM cell
// rsrp       Reference Signal Received Power [dBm]

#define TrM2M_PosUpdateLTE [.type, .stamp, .mcc, .mnc, .lac, .cid, .rsrp]
```

TNMEA_GGA

Information (position, height above sea level and accuracy) extracted from a GGA data record

```
// Lat        geographical latitude in degrees (resolution: 0.000001°)
//            -90,000,000 = South pole 90° S,
//                      0 = Equator,
//            +90,000,000 = North pole 90° N
//
// Long       geographical longitude in degrees (resolution: 0.000001°)
//            -180,000,000 =180° West, 0 =Zero meridian, +180,000,000 =180° East
//
// Alt        Height above sea level in meters
// Qual       NMEA Quality indicator(see "Constants" on page 182)
// SatUsed    Number of satellites used for the positioning
// HDOP       relative accuracy of the horizontal position [0,01]

#define TNMEA_GGA[.Lat, .Long, .Alt, .Qual, .SatUsed, .HDOP]
```

15.2.8.2 Constants

List of the supported types of cell/network information entries

Possible types of cell/network information entries that can be read from the system via the function "rM2M_EnumPosUpdate()"

```
RM2M_POSUPDATE_TYPE_ERR  = 0, //invalid entry
RM2M_POSUPDATE_TYPE_GSM  = 1, //Information about a GSM cell
RM2M_POSUPDATE_TYPE_UMTS = 2, //Information about a UMTS cell
RM2M_POSUPDATE_TYPE_LTE  = 3, //Information about an LTE cell
RM2M_POSUPDATE_TYPE_WIFI = 4, //Information about a WiFi network
```

NMEA error codes

Error codes of the function rM2M_SetPosNMEA()

```
RM2M_NMEA_ERR_DATATYPE  = -2, // Data type (e.g. $GGSA) not supported.
RM2M_NMEA_ERR_SENTENCE  = -3, // Sentence invalid (e.g. checksum error)
RM2M_NMEA_ERR_LATITUDE  = -4, // Geographical latitude invalid
RM2M_NMEA_ERR_LONGITUDE = -5, // Geographical longitude invalid
RM2M_NMEA_ERR_ALTITUDE  = -6, // Altitude above sea level invalid
RM2M_NMEA_ERR_SAT_USED  = -7, // Number of satellites used invalid.
RM2M_NMEA_ERR_QUAL      = -8, // GPS quality indication not supported.
```

NMEA quality indicator

```
RM2M_NMEA_FIX_NOK      = 0,   // invalid/no fix
RM2M_NMEA_FIX_GPS      = 1,   // Non-differential GPS fix
RM2M_NMEA_FIX_DGPS     = 2,   // Differential GPS fix
RM2M_NMEA_FIX_PPS      = 3,   // Precise positioning service (PPS)
RM2M_NMEA_FIX_RTK      = 4,   // Real time kinematic (RTK)
RM2M_NMEA_FIX_FLOATRTK = 5,   // Float real time kinematic
RM2M_NMEA_FIX_EST      = 6,   // Estimated fix (dead reckoning, coupled
                              // navigation)
RM2M_NMEA_FIX_MAN      = 7,   // Manual input mode
RM2M_NMEA_FIX_SIM      = 8,   // Simulation mode
```

List of supported GNSS device IDs

Designed to identify the source of the NMEA data record (in accordance with the "Talker ID" used with the NMEA 0183 standard)

```
RM2M_NMEA_DEVICE_GP = 0x244750,  // $GP (GPS)
RM2M_NMEA_DEVICE_GL = 0x24474C,  // $GL (GLONASS)
RM2M_NMEA_DEVICE_GA = 0x244741,  // $GA (GALILEO)
RM2M_NMEA_DEVICE_GN = 0x24474E,  // $GN (GENERIC GNSS)
```

List of supported NMEA data records

```
RM2M_NMEA_RECORD_GGA = 0x474741, // GGA (global positioning system fix data)
```

15.2.8.3 Functions

native rM2M_SetPos(Lat, Long, Elev, Qual, SatUsed);

Saves the GPS position information in the device. A historical record is not maintained. This means that the current position information always overwrites the last known position. The information is transmitted to the myDatanet server and can, for example, be read out via the API (see "API" on page 263).

Parameter	*Explanation*
Lat	*geographical latitude in degrees (resolution: 0.000001°)* *-90 000 000 = South pole 90° south* *0 = Equator* *+90 000 000 = North pole 90° north*
Long	*geographical longitude in degrees (resolution: 0.000001°)* *-180 000 000 = 180° west* *0 = Zero meridian (Greenwich)* *+180 000 000 = 180° east*
Elev	*Height above sea level in meters (Valid range: -999...+9999)*
Qual	*Quality indicator (GPS quality indicator)* *RM2M_NMEA_FIX_NOK: invalid/no fix* *RM2M_NMEA_FIX_GPS: non-differential GPS fix* *RM2M_NMEA_FIX_DGPS: differential GPS fix* *RM2M_NMEA_FIX_PPS: Precise positioning service (PPS)* *RM2M_NMEA_FIX_RTK: Real time kinematic (RTK)* *RM2M_NMEA_FIX_FLOATRTK: Float real time kinematic* *RM2M_NMEA_FIX_EST: Estimated fix (dead reckoning, coupled navigation)* *RM2M_NMEA_FIX_MAN: Manual input mode* *RM2M_NMEA_FIX_SIM: Simulation mode*
SatUsed	*Number of satellites used for the positioning (valid range: 0 ... 99)*

	Explanation
Return value	• *OK, if successful* • *ERROR* ***Note:** The parameters are checked against the specified range limits. The function returns "ERROR" if the limits are not adhered to.*

native rM2M_DecodeNMEA(const sentence{}, data[], len=sizeof data);

Decodes a transferred NMEA data record

Parameter	*Explanation*
sentence	*NMEA data record from a GPS receiver starting with the '$' character.* ***Important note:** The strings must be terminated ('\0') immediately after the checksum.*
data	*Buffer (cell array) to store the decoded data* *[0] : Contains the GNSS device ID (see "List of supported GNSS device IDs" in chapter "Constants" on page 182)* *[1] : Contains the type of decoded NMEA data record (see "List of supported NMEA data records" in chapter "Constants" on page 182)* *[2] ... [n] : Dependent on type of decoded NMEA data record* *For a type "RM2M_NMEA_RECORD_GGA" data record, the remaining structure is the same as the "TNMEA_GGA" structure.* *[2]: .Lat* *[3]: .Long* *[4]: .Alt* *[5]: .Qual* *[6]: .SatUsed* *[7]: .HDOP*
len	*Size (in cells) of the buffer to record the decoded data – OPTIONAL*

	Explanation
Return value	• *positive value, if successful (number of filled array elements, i.e. cells)* • *negative value, if an error has occurred (see "NMEA error codes" in chapter "Constants" on page 182)*

native rM2M_SetPosNMEA(const Sentence{});

Takes the GPS position information from the transferred NMEA data record and saves it in the device. A historical record is not maintained. This means that the current position information always overwrites the last known position. The information is transmitted to the myDatanet server and can, for example, be read out via the API (see "API" on page 263).

Parameter	***Explanation***
Sentence	*NMEA data record from a GPS receiver starting with the '$' character. The following data records are currently supported:* • *$GPGGA - location specification (fix information)* ***Important note:*** *The strings must be terminated ('\0') immediately after the checksum.*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "NMEA error codes" in chapter "Constants" on page 182)*

native rM2M_GetPos(&Lat, &Long, &Elev, &Qual=0, &SatUsed=0);

Reads out the GPS position information saved to the device

Parameter	*Explanation*
Lat	*Variable to store the geographical latitude in degrees (resolution: 0.000001°)* *-90 000 000 = South pole 90° south* *0 = Equator* *+90 000 000 = North pole 90° north*
Long	*Variable to store the geographical longitude in degrees (resolution: 0.000001°)* *-180 000 000 = 180° west* *0 = Zero meridian (Greenwich)* *+180 000 000 = 180° east*
Elev	*Variable to store the height above sea level in metres (valid range: -999...+9999)*
Qual	*Variable to store the quality indicator (GPS quality indicator) – OPTIONAL* *RM2M_NMEA_FIX_NOK: invalid/no fix* *RM2M_NMEA_FIX_GPS: non-differential GPS fix* *RM2M_NMEA_FIX_DGPS: differential GPS fix* *RM2M_NMEA_FIX_PPS: Precise positioning service (PPS)* *RM2M_NMEA_FIX_RTK: Real time kinematic (RTK)* *RM2M_NMEA_FIX_FLOATRTK: Float real time kinematic* *RM2M_NMEA_FIX_EST: Estimated fix (dead reckoning, coupled navigation)* *RM2M_NMEA_FIX_MAN: Manual input mode* *RM2M_NMEA_FIX_SIM: Simulation mode*
SatUsed	*Variable to store the number of satellites used for positioning – OPTIONAL*

	Explanation
Return value	• *OK if valid GPS position information is stored in the device* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native rM2M_EnumPosUpdate(...);

lists the information saved in the device about the GSM/UMTS/LTE cells and WiFi networks in the receiving range. With this function a variable list of parameters is used. The parameters to be passed depend on the purpose. The following procedure is recommended:

1. *Reading of the number of available cell/network information entries*

```
new nEnum;

rM2M_EnumPosUpdate(nEnum);
```

2. *Determination of the particular type of the cell/network information entries*

```
new type;
new idxEnum = 0;

for(idxEnum=0 ; idxEnum < nEnum ; idxEnum++)
   rM2M_EnumPosUpdate(idxEnum, type);
```

3. *Reading of cell/network information entries based on the types determined previously (in the following example only those that contain information about a GSM cell).*

```
new sGSMPos[TrM2M_PosUpdateGSM];

if(type == RM2M_POSUPDATE_TYPE_GSM)
   rM2M_EnumPosUpdate(idxEnum, sGSMPos, sizeof sGSMPos);
```

Parameter	*Explanation*
nEnum	*Variable to store the number of available cell/network information entries*
idxEnum	*Index of the cell/network information entry whose type should be determined or that should be read by the system.* *Either the "type" parameter or the two "buf" and "len" parameters are required in addition depending on the desired action.*
type	*Variable to store the type of a cell/network information entry (see "RM2M_ POSUPDATE_TYPE_xxx" in Chapter "Constants" on page 182)*
buf	*Buffer to store a cell/network information entry* *The structure of the buffer depends on the cell/network information entry to be read (see "TrM2M_PosUpdatexxx" in Chapter "Arrays with symbolic indices" on page 180)*
len	*Size (in cells) of the structure to store a cell/network information entry*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native rM2M_GetGSMPos(posidx, pos[TrM2M_GSMPos]=0);

Returns the number of GSM/UMTS/LTE cells for which valid information is saved to the device (posidx < 0) or reads out the information saved to the device about a GSM/UMTS/LTE cell in the receiving range (posidx >= 0)

> ***Note:*** *Use the "rM2M_EnumPosUpdate()" function in order to get information on WiFi networks in the receiving range or more specific information on UMTS and/or LTE cells.*

Parameter	***Explanation***
posidx	*Selection of the information returned by the function* *posidx < 0:* *Read the number of GSM/UMTS/LTE cells for which valid information is saved to the device* *posidx >=0:* *Number of the GSM/UMTS/LTE cell information block that should be read*
pos	*posidx < 0:* *Not required* *posidx >=0:* *Structure for storing the information about a GSM/UMTS/LTE cell in the receiving range (see "TrM2M_GSMPos" in chapter "Arrays with symbolic indices" on page 180)*

	Explanation
Return value	*posidx < 0:* *Number of GSM/UMTS/LTE cells for which valid information is saved to the device (max. 10)* *posidx >=0:* • *OK, if the desired cell information block contains valid data of a GSM cell* • *OK+1, if the desired cell information block contains valid data of a UMTS cell* • *OK+2, if the desired cell information block contains valid data of a LTE cell* • *ERROR*

15.2.9 Math

Helpful constants

Definition	Value	Description
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	$\log_2 e$
M_LOG10E	0.43429448190325182765	$\log_{10} e$
M_LN2	0.69314718055994530942	ln 2
M_LN10	2.30258509299404568402	ln 10
M_PI	3.14159265358979323846	π
M_PI_2	1.57079632679489661923	π/2
M_PI_4	0.78539816339744830962	π/4
M_1_PI	0.31830988618379067154	1/π
M_2_PI	0.63661977236758134308	2/π
M_2_SQRTPI	1.12837916709551257390	2/sqrt(π)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)

native fround(Float:x);

Commercially rounds the transferred float

Parameter	*Explanation*
x	*Float that should be rounded*

	Explanation
Return value	*Commercially rounded integral value*

native min(value1, value2);

Supplies the smaller of the two transferred values

Parameter	*Explanation*
value1	*Two values of which the smaller one is to be determined*
value2	

	Explanation
Return value	*The smaller of the two transferred values*

native max(value1, value2);

Supplies the larger of the two transferred values

Parameter	*Explanation*
value1	*Two values of which the larger one is to be determined*
value1	

	Explanation
Return value	*The larger of the two transferred values*

native clamp(value, min=cellmin, max=cellmax);

Checks whether the transferred value is between "min" and "max"

Parameter	*Explanation*
value	*Value that is to be checked*
min	*Lower limit*
max	*Upper limit*

	Explanation
Return value	• *"value" if the value is between "min" and "max"* • *"min" is the value is less than "min"* • *"max", if the value is greater than "max"*

native swapchars(c);

Swaps the order of the bytes

Parameter	*Explanation*
c	*Value for which the bytes should be swapped over*

	Explanation
Return value	*Value for which the bytes in parameter "c" are swapped over (the lowest byte becomes the highest byte)*

The mode of operation of the following functions corresponds to that of the standard ANSI-C implementation:

native Float:sin(Float:x);

Sine of x

native Float:cos(Float:x);

Cosine of x

native Float:tan(Float:x);

Tangent of x

native Float:asin(Float:x);
Arcsine(x) in the range [-π/2, π/2], x element of [-1, 1]

native Float:acos(Float:x);
Arccosine(x) in the range [0, π], x element of [-1, 1]

native Float:atan(Float:x);
Arctangent(x) in the range [-π/2, π/2]

native Float:atan2(Float:y, Float:x);
Arctangent(y/x) in the range [-π, π]

native Float:sinh(Float:x);
Hyperbolic sine of x

native Float:cosh(Float:x);
Hyperbolic cosine of x

native Float:tanh(Float:x);
Hyperbolic tangent of x

native Float:exp(Float:x);
Exponential function e^x

native Float:log(Float:x);
Natural logarithm ln(x), x > 0

native Float:log10(Float:x);
Logarithm as the basis 10 $\log_{10}(x)$, x > 0

native Float:pow(Float:x, Float:y);
x^y. An argument error has occurred if x = 0 and y <= 0, or if x < 0 and y is not a whole number.

native Float:sqrt(Float:x);
Square root x, x >= 0

native Float:ceil(Float:x);
Smallest whole number that is not smaller than x

native Float:floor(Float:x);
Largest whole number that is not larger than x

native Float:fabs(Float:x);
Absolute value | x |

native Float:ldexp(Float:x, n);
$x*2^n$

native Float:frexp(Float:x, &n);
Breaks down x into a normalised mantissa in the range [1/2, 1] that is supplied as the result, and a potency of 2 that is filed in n. If x is zero, both parts of the result are zero.

native Float:modf(Float:x, &Float:ip);
Breaks down x into an integral and residual part that both have the same prefix as x. The integral part is filed in ip, while the residual part is the result.

native Float:fmod(Float:x, Float:y);
Residual floating point of x/y with the same prefix as x. The result is dependent on the implementation, if y is zero.

native isnan(Float:x);
Returns a value that is not equal to zero, if x is not a number

15.2.10 Char & String

The mode of operation of the following functions essentially corresponds to that of the standard ANSI-C implementation:

native strlen(const string[]);

Returns the length of the string (without '\0')

Parameter	*Explanation*
string	*Character string for which the length has to be determined*

	Explanation
Return value	*Number of characters without the final '\0'*

native sprintf(dest[], maxlength=sizeof dest, const format[], {Float,Fixed,_}:...);

Saves the transferred format string in the array dest. The mode of operation of the functions corresponds to that of the "snprintf" function of the standard ANSI-C implementation.

Note:

- *If resulting string is longer than <dest>'s size, the very last character is set to terminating zero.*
- *<dest> size is always rounded up to full multiple of 4.*

Parameter	***Explanation***
dest	*Array to store the formatted result*
maxlength	*Maximum number of characters that the array dest can store*
format	*The format character string to be used (C-style formatting codes)* *%b : Number in binary radix* *%c : Character* *%d : Number in decimal radix* *%f : Floating point number* *%s : String* *%x : Number in hexadecimal radix* *... : s32 | f32 | astr - Additional arguments.* *Depending on the format string, the function may expect a sequence of additional arguments, each containing a value to be used to replace a format specifier in the format string.*

	Explanation
Return value	• *-1 in the event of a fault* • *Number of characters that would have been written if the array dest had been long enough (without '\0').* *The array dest is always assigned a final zero. The length of the array dest cannot be exceeded.*

native strcpy(dest[], const source[], maxlength=sizeof dest);

Copies the source character string to the array dest (including '\0').

Parameter	***Explanation***
dest	*Array to store the character string that should be copied*
source	*Character string that should be copied*
maxlength	*Size (in cells) of the array to store the character string to be copied - OPTIONAL*

	Explanation
Return value	*Number of copied characters*

native strcat(dest[], const source[], maxlength=sizeof dest);

Adds the source character string to the dest character string (including '\0')

Important note: *Both strings must be zero-terminated.*

Parameter	***Explanation***
dest	*Array to store the result. This array already contains one character string to which the source character string should be added.*
source	*Character string that should be added to the character string included in the array dest*
maxlength	*Size (in Cells) of the array to store the result - OPTIONAL*

	Explanation
Return value	*Number of added characters*

native strcmp(const string1[], const string2[], length=cellmax);

Compares character string1 and string2

Important note: *Both strings must be zero-terminated.*

Parameter	***Explanation***
string1	*The two character strings that are to be compared*
string2	
length	*The maximum number of characters that should be taken into consideration during the comparison - OPTIONAL*

	Explanation
Return value	• *1: string1 > string 2* • *0: both of the character strings are the same (at least the length that is taken into account)* • *-1: string1 < string 2*

native strchr(const string[], char);

Searches for a character (first occurrence) in a character string

Parameter	***Explanation***
string	*Character string that should be searched*
char	*Character that the search is looking for*

	Explanation
Return value	• *-1, if the character that the search is looking for is not included in the character string* • *Array index of the character that the search is looking for (first character occurring in the character string)*

native strrchr(const string[], char);

Searches for a character (last occurrence) in a character string

Parameter	*Explanation*
string	*Character string that should be searched*
char	*Character that the search is looking for*

	Explanation
Return value	• *-1, if the character that the search is looking for is not included in the character string* • *Array index for the character that the search is looking for (last character occurring in the character string)*

native strspn(const string1[], const string2[]);

Searches for the position of the first character in string1 that is ***not****included in the character string of permitted characters (string2)*

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *Length of string1 if no forbidden characters are found* • *Position of the first character in the character string that should be searched that is not included in the character string of permitted characters*

native strcspn(const string1[], const string2[]);

Searches for the position of the first character in string1 that is also included in the character string of permitted characters (string2)

Note: *See similar function strpbrk () which has a slightly different result.*

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *Length of string1 if no permitted character has been found* • *Position of the first character in the character string that should be searched that is also included in the character string of permitted characters*

native strpbrk(const string1[], const string2[]);

Searches the array index of the first character that is also included in the character string of permitted characters

Note: *See similar function strcspn () which has a slightly different result.*

Parameter	***Explanation***
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *-1: If no permitted character has been found* • *>=0: Array index of the first character in the character string that should be searched that is also included in the character string of permitted characters*

native strstr(const string1[], const string2[]);

Searches character string2 in character string1

Parameter	***Explanation***
string1	*Character string to search in*
string2	*Character string to search for*

	Explanation
Return value	• *-1: if character string2 that is being searched for is not included in string1* • *>=0: Array index where character string2 that is being searched for starts in string1*

native strtol(const string[], base);

Converts a character string into a value

Note:

- *Function differs slightly from it's C variant.*
- *Parsing consumes as many characters as possible, up to the first char not matching with given base.*

Parameter	***Explanation***
string	*Character string to be converted* ***Important note:*** *Strings > 128 bytes are not supported!*
base	*Specifies the basis that must be used for the conversion* *2-36: The specified basis is used* *0: 8, 10 or 16 is used as the basis, depending on the character string to be converted* *Basis 8: with a leading 0* *Basis 16: with 0x or 0X* *Base 10: default*

	Explanation
Return value	*Value that corresponds to the character string*

native Float: atof(const string[]);

Converts a character string into a float

Note:

- *Decimal separator is always ".", no thousands separators supported.*
- *Parsing consumes as many characters as possible, up to the first none-float char.*

Parameter	***Explanation***
string	*Character string to be converted* ***Important note:*** *Strings > 128 bytes are not supported!*

	Explanation
Return value	*Float for which the numerical value corresponds to the character string*

native memcpy_native(dst{}, const dstofs, const src{}, const srcofs, const bytes, const dst_cells=sizeof dst, const src_cells=sizeof src);

Copies bytes from one buffer to another one

Parameter	*Explanation*
dst	*Target buffer to which the data should be copied*
dstofs	*Position (byte offset) in the target buffer to which the data should be copied*
src	*Source buffer from which the data should be copied*
srcofs	*Position (byte offset) within the source buffer from which the data should be copied*
bytes	*Number of bytes that should be copied*
dst_cells	*Size (in cells) of the target buffer - OPTIONAL*
src_cells	*Size (in cells) of the source buffer - OPTIONAL*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, if one of the following errors occurs*
	  • *If one of the two byte offsets or the number of bytes to be copied is < 0*
	  • *If the byte offsets refer to a byte outside the relevant buffer*
	  • *If the reading was to exceed the source buffer (i.e. "Source byte offset" + "Number of bytes" would refer to a byte outside the source buffer)*
	  • *If the reading was to exceed the target buffer (i.e. "Target byte offset" + "Number of bytes" would refer to a byte outside the target buffer)*
	  • *If invalid buffers were transferred*

native memset_native(dst{}, const dstofs, const srcval, const bytes, dstcells=sizeof dst);

Writes the desired value into the individual bytes of the transferred buffer

Parameter	*Explanation*
dst	*Buffer in which the bytes should be set to the desired value*
dstofs	*Position (byte offset) within the transferred buffer from which the bytes should be set to the desired value*
srcval	*Value to which the individual bytes should be set*
bytes	*Number of bytes that should be set to the desired value*
dstcells	*Size (in cells) of the transferred buffer - OPTIONAL*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, if one of the following errors occurs*
	  • *The byte offset is < 0*
	  • *The byte offset refers to a byte outside the buffer*
	  • *If an invalid buffer was transferred*

native memcmp_native(const src1{}, const src1ofs, const src2{}, const src2ofs, bytes, src1cells=sizeof src1, src2cells=sizeof src2);

Compares two buffers, byte for byte

Parameter	*Explanation*
src1	*Buffer #1*
src1ofs	*Position (byte offset) within buffer #1 from which the bytes should be compared*
src2	*Buffer #2*
src2ofs	*Position (byte offset) within buffer #2 from which the bytes should be compared*
bytes	*Number of bytes that should be compared*
src1cells	*Size (in cells) of buffer #1 - OPTIONAL*
src2cells	*Size (in cells) of buffer #2 - OPTIONAL*

	Explanation
Return value	• *> 0: Buffer #1 > Buffer #2* • *0: the content of both of the buffers is the same (at least the bytes that are taken into account)* • *<0: Buffer #1 < Buffer #2*

native tolower(c);

Converts a character into lower case

Parameter	*Explanation*
c	*Character that should be converted to lower case*

	Explanation
Return value	*The lower case variant of the transferred character, if available, or the unchanged character code of "c" if the letter "c" does not have a lower case equivalent.*

native toupper(c);

Converts a character into upper case

Parameter	*Explanation*
c	*Character that should be converted to upper case*

	Explanation
Return value	*The upper case variant of the transferred character, if available, or the unchanged character code of "c" if the letter "c" does not have a upper case equivalent.*

15.2.11 CRC & hash

15.2.11.1 Arrays with symbolic indices

TMD5_Ctx

Context structure for the MD5 calculation

```
// init    After being set to "0", the context structure can be used to
//         calculate a new hash. If a calculation should be implemented by
//         calling up the "MD5" function repeatedly, there must not be
//         write access to this element.
// tmp     no write access permitted, for internal use

#define TMD5_Ctx[.init, .tmp[22]]
```

15.2.11.2 Functions

native CRC16(data{}, len, initial=0xFFFF);

Returns the calculated modbus CRC16 of the transferred data

Parameter	*Explanation*
data	*Array that contains the data for which the CRC16 should be calculated*
len	*Number of bytes that must be taken into consideration during the calculation*
initial	*Initial value for calculating the CRC16 - OPTIONAL*

	Explanation
Return value	*Calculated CRC16*

native CRC32(data{}, len, initial=0);

Returns the calculated Ethernet CRC32 of the transferred data

Parameter	*Explanation*
data	*Array that contains the data for which the CRC32 should be calculated*
len	*Number of bytes that must be taken into consideration during the calculation*
initial	*Initial value for calculating the CRC32 - OPTIONAL*

	Explanation
Return value	*Calculated CRC32*

native MD5(data{}, len, hash{16}, ctx[TMD5_Ctx] = [0]);

Calculates the MD5 hash for the transferred data. If the hash for a data block should be calculated by calling up this function several times (e.g. when receiving data in blocks), then the same context structure must be transferred every time the function is called up. The context structure must not be changed between function call-ups. If the hash can be calculated by calling up the function once (e.g. complete data block is already available), then it is not necessary to transfer its own context structure.

Parameter	*Explanation*
data	*Array that contains the data for which the MD5 hash should be calculated*
len	*Number of bytes that must be taken into consideration during the calculation*
hash	*Array to store the calculated 128-bit hash value*
ctx	*Context structure for the MD5 calculation – OPTIONAL (required only if calculation uses multiple calls to MD5())*

	Explanation
Return value	*---*

15.2.12 Various

15.2.12.1 Arrays with symbolic indices

TablePoint

Two-column reference point table, integer data type

```
// key     Column that is searched
// value   Column with the result values that need to be returned

#define TablePoint[.key, .value]
```

TablePointF

Two-column reference point table, float data type

```
// key     Column that is searched
// value   Column with the result values that need to be returned

#define TablePointF[Float:.key, Float:.value]
```

TrM2M_Id

Information for identifying the module/device

```
// string     rapidM2M module identification (e.g. "rapidM2M EasyIoT HW1.1")
// module     rapidM2M module type (e.g. "EasyIoT")
// hwmajor    Hardware: Major version number
// hwminor    Hardware: Minor version number
// sn         Device serial number (binary) in BIG endian format
//            E.g.: "010146AF251CED1C" --> "01" in sn{0}, "1C" in sn{7}
// fwmajor    Firmware: Major version number
// fwminor    Firmware: Minor version number
// ctx        Site title (context)
//            Empty string if no context available

#define TrM2M_Id[  .string{50}, .module{10}, .hwmajor, .hwminor,
                   .sn{8}, .fwmajor, .fwminor, .ctx{50}             ]
```

TRTM_Data

Information regarding the runtime measurement

```
// runtime       Determined runtime in [ms]
// instructions  Number of executed instructions
// tmp           For internal use, no write access permitted

#define TRTM_Data[.runtime, .instructions, .tmp[3]]
```

15.2.12.2 Constants

Error codes for the "CalcTable" and "CalcTableF" functions

```
const
{
  TAB_ERR_FLOOR = -1, // searched value lower than the first table entry
  TAB_ERR_CEIL = -2,  // searched value higher than the last table entry
};
```

15.2.12.3 Functions

native getapilevel();

Issues the implemented API level of the script engine

	Explanation
Return value	*Implemented API level of the script engine* • *1: Initial functionality* • *2: Added function "exists ()" to check availability of runtime function* • *3: Added function "loadmodule ()" to load a device logic module*

native exists(const name[]);

checks whether the required rapidM2M API function is supported by the device firmware

> ***Important note:*** *Use getapilevel () upfront to check if exists() is avilable with your firmware version.*

Parameter	***Explanation***
name	*Name of the required rapidM2M API function*

	Explanation
Return value	• *true, if the function is available* • *false, if the device firmware does not support the function*

native loadmodule(mod{});

Loads a script module at the runtime. This enables the script engine to be extended by its own native functions. The implementation of operations as a native function means that processing speeds can be increased significantly in comparison to the implementation in script. A script module can contain several native functions. After calling up this function, the native functions contained in the script module can be used in the same way as the standard functions available in the script engine.

Parameter	***Explanation***
mod	*Byte array that contains the script module to be loaded.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native rtm_start(measurement[TRTM_Data]);

Starts a runtime measurement

> ***Important note:*** *Execution of concurrent measurements is not allowed.*

Parameter	***Explanation***
measurement	*Structure for storing the information regarding a runtime measurement* ***Important note:*** *This structure must be persistent from calling up "rtm_start()" to calling up "rtm_stop()".*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native rtm_stop(measurement[TRTM_Data]);

Stops the runtime measurement and calculates the time in [ms] since the "rtm_start()" function was called up and the instructions executed since then. The determined values are written in the ".runtime" and ".instructions" elements of the transferred structure to record the information regarding a runtime measurement.

Parameter	***Explanation***
measurement	*Structure for storing the information regarding a runtime measurement* ***Important note:*** *This structure must be persistent from calling up "rtm_start()" to calling up "rtm_stop()".*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

native CalcTable(key, &value, const table[][TablePoint], size = sizeof table);

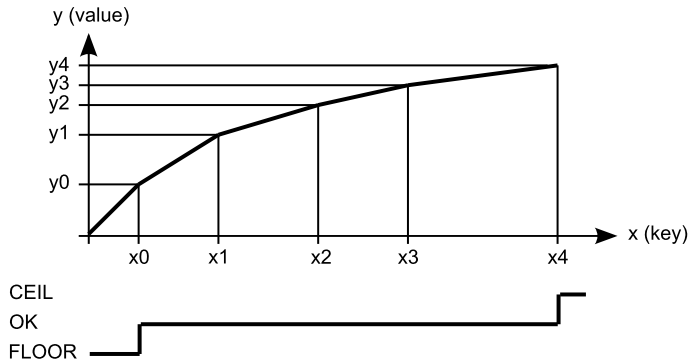
*Searches for a certain value in the "key" column of the transferred reference point table and supplies the relevant value from the "value" column in the table. If the searched value is between two reference points, the returned value is interpolated linearly between the two adjacent values in the "value" column (linear equation: y = k*x + d). Non-linear characteristic curves (e.g. connection between ADC value -> temperature) can be reproduced with this function.*

Parameter	*Explanation*
key	*Value that is used for the search*
value	*Includes the result of the calculation by the function*
table	*The table that is searched must be a "TablePoint" type table.*
size	*Number of rows in the table*

	Explanation
Return value	• *OK, if the relevant value was found* • *TAB_ERR_FLOOR, if the searched value is lower than the first table entry. "value" contains the first table entry.* • *TAB_ERR_CEIL, if the searched value is higher than the last table entry. "value" contains the last table entry.* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).*

Note: *Additional explanation on the "`table`" reference point table*

The rows of the table can be displayed in an x/y coordinate system. The values in the "key" column are displayed on the X axis and the associated values in the "value" column are displayed on the Y axis.



Display of the reference point table as an x/y coordinate system

native CalcTableF(Float:key, &Float:value, const table[][TablePointF], size = sizeof table);

The functionality is the same as that of the "CalcTable" function. The difference is that "Float" is the data type for all elements of the "CalcTableF" function.

native rM2M_GetId(id[TrM2M_Id], len=sizeof id);

Provides the information to identify the module/device

Parameter	*Explanation*
id	*Structure for storing the information to identify the module/device (see "TrM2M_Id" in chapter "Arrays with symbolic indices" on page 201)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation
Return value	• *Used size (in cells) of the structure for storing the information* • *ERROR if the address and/or length of the ID structure are invalid (outside the script data memory)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132).* ***Note:** The firmware of a module/device detects if a device logic is being used for which the function only has one transfer parameter (older include file is being used) and for compatibility reasons therefore returns "OK" instead of the size of the structure for storing the information.*

native heapspace();

Supplies the free memory capacity to the heap

	Explanation
Return value	*The free memory capacity to the heap. The stack and the heap have a joint memory area, so that this value specifies the number of bytes that remain for the stack or the heap.*

native funcidx(const name[]);

Supplies the index of a public function. Used to register callbacks for the runtime environment.

Parameter	*Explanation*
name	*Name of the public function*

	Explanation
Return value	• *-1, if there is no function with the transferred name* • *Index of the public function*

native numargs();

Returns the number of arguments transferred to a function. This is useful within functions with a variable list of arguments.

	Explanation
Return value	*The number of arguments that have been transferred to a function.*

native getarg(arg, index=0);

This function supplies an argument from a variable argument list. If the argument is an array, the "index" specifies the index of the required array element.

Parameter	*Explanation*
arg	*The sequence number of the argument. Use 0 for the first argument.*
index	*Index if "arg" refers to an array*

	Explanation
Return value	*The value of the argument.*

native setarg(arg, index=0, value);

Sets the value of the argument

Parameter	*Explanation*
arg	*The sequence number of the argument. Use 0 for the first argument.*
index	*Index if "arg" refers to an array*
value	*Value to which the argument should be set*

	Explanation
Return value	• *true, if the value could be set* • *false, if the argument or index are invalid* *This function sets an argument in a variable argument list. If the argument is an array, the "index" specifies the index of the required array element.*

native rand();

Returns a random number from the "32-Bit signed Integer" value range. However, value "-1" (ERROR) is reserved for returning an error.

	Explanation
Return value	• *Random number from the "32-Bit signed Integer" value range* • *ERROR if no random number generator is available*

native delay_us(us);

Blocking delay function. The execution of the device logic is stopped and the following code line is only executed once the delay time has expired.

Parameter	*Explanation*
us	*Delay time (1...10000 [µs]).*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an error occurs*

15.2.13 Console

native print(const string[]);

Prints the specified string to the standard output

Parameter	*Explanation*
string	*The character string to be issued. This can include escape sequences.*

	Explanation
Return value	*OK*

native printf(const format[], {Float,Fixed,_}:...);

Prints the transferred format string to the standard output. The mode of operation of the functions corresponds to that of the standard ANSI-C implementation.

Note:

- *Characters may get lost if console output buffer overflows.*
- *Use sprintf () to write to a string buffer instead of the console.*

Parameter	***Explanation***
format[]	*The format character string to be used (C-style formatting codes)* *%b : Number in binary radix* *%c : Character* *%d : Number in decimal radix* *%f : Floating point number* *%s : String* *%x : Number in hexadecimal radix* *... : s32 | f32 | astr - Additional arguments.* *Depending on the format string, the function may expect a sequence of additional arguments, each containing a value to be used to replace a format specifier in the format string.*

	Explanation
Return value	• *Number of printed characters* • *ERROR, if not successful*

native setbuf(buf{}, size);

Provides the firmware with a buffer from the RAM area reserved for the device logic that is used to output strings via the "printf()" function. When this function is called up, the system switches from the 256 byte buffer integrated in the firmware to the transferred buffer.

***Important note:** The buffer must be valid during the entire use by the firmware (i.e. it must be defined as a global or static variable).*

Parameter	***Explanation***
buf	*Static byte array that should be used as a buffer to output strings*
size	*Size of the buffer in bytes* ***Note:** If the function is called up again and the size is set to "0" during the process, then the system switches back to the integrated buffer (256 bytes). The transferred static byte array can then be used by the device logic again.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if not successful*

15.2.14 SMS

Important note: *If the device is in "online" mode no SMS can be processed.*

15.2.14.1 Callback functions

public func(const SmsTel[], const SmsText[]);

Function to be provided by the device logic developer, that is called up if an SMS is received

Parameter	*Explanation*
SmsTel	*String that contains the telephone number of the sender of the SMS*
SmsText	*String that contains the content of the SMS*

15.2.14.2 Functions

native rM2M_SmsInit(funcidx, config);

Initialises SMS receipt

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up if an SMS has been received* *Type of function: public func(const SmsTel[], const SmsText[]);* ***Important note:*** *If an SMS that is longer than 160 characters is received, it is discarded immediately. The specified public function is not called up.*
config	*Reserved for extensions*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native rM2M_SmsClose();

Deactivates SMS receipt

	Explanation
Return value	*OK*

15.2.15 External SIM

15.2.15.1 Arrays with symbolic indices

TrM2M_SIMCfg

Configuration data of an external SIM card

```
// pin         Pin code for the external SIM card
// apn         Access point (APN) that should be used for the connection
// username    Username to dial up via the access point
// password    Password to dial up via the access point

#define TrM2M_SIMCfg[.pin{8}, .apn{40}, .username{40}, .password{40}]
```

15.2.15.2 Functions

native rM2M_SetExtSimCfg(cfg[TrM2M_SIMCfg], len=sizeof cfg);

Saves the transferred configuration data for the external SIM card in the device. Setting the configuration data switches to the external SIM card. To switch back to the internal SIM chip, a structure, in which all of the fields are set to 0, must be transferred when setting the configuration data.

> ***Note:*** *The chargeable feature "Activation code VPN SIM (300539)" must be released to be able to use the external SIM card.*

Parameter	*Explanation*
cfg	*Structure that contains the configuration data for the external SIM card (see "TrM2M_SIMCfg" in chapter "Arrays with symbolic indices" on page 211)*
len	*Size (in cells) of the structure that contains the configuration data - OPTIONAL*

	Explanation
Return value	• *Used size (in cells) of the structure for storing the configuration data* • *ERROR if the address and/or length of the info structure are invalid (outside the script data memory)*

native rM2M_GetExtSimCfg(cfg[TrM2M_SIMCfg]=0, len=sizeof cfg);

Provides the current configuration data stored for the external SIM card

Parameter	*Explanation*
cfg	*Structure to store the configuration data saved for the external SIM card (see "TrM2M_SIMCfg" in chapter "Arrays with symbolic indices" on page 211)*
len	*Size (in cells) of the structure to store the saved configuration data - OPTIONAL*

	Explanation

Return value	• *Used size (in cells) of the structure for storing the configuration data*
	• *ERROR, if one of the following errors occurs*
	  ◦ *Address and/or length of the cfg structure is invalid (outside the script data memory)*
	  ◦ *No valid configuration data available for the external SIM card*

15.2.16 File transfer

15.2.16.1 Arrays with symbolic indices

TFT_Info

Properties of a file entry

```
// name       Name of the file
// stamp      Time stamp of the file (seconds since 31.12.1999)
// stamp256   Fraction of the next started sec. (resolution 1/256 sec.)
// size       File size in byte
// crc        Ethernet CRC32 of the file
// flags      File flags (see "File flags" in
//            chapter "Constants" on page 212)
#define TFT_Info[  .name{256}, .stamp, .stamp256, .size, .crc, .flags ]
```

15.2.16.2 Constants

File flags

```
FT_FLAG_READ   = 0x0001,   // File can be read by the server.
FT_FLAG_WRITE  = 0x0002,   // File can be written by the server.
FT_FLAG_NODE   = 0x0004    /* file nodes (required to entitle the server to
                              create a new file) */
FT_FLAG_SYSTEM = 0x0008    // System file (cannot be used by the device logic)
```

File transfer command

```
FT_CMD_NONE   = 0,
FT_CMD_UNLOCK = 1,         /* File transfer session terminated. The server
                              releases the block again. */
FT_CMD_LIST   = 2,         // The server requests the properties of a file
FT_CMD_READ   = 3,         // The server requests a part of a file.
FT_CMD_STORE  = 4,         // The server requests a file to be written.
FT_CMD_WRITE  = 5,         /* The server provides a block to be written in
                              a file. */
FT_CMD_DELETE = 6,         // The server requests a file to be deleted.
FT_CMD_ENUM   = 7,         /* The server requests the properties of a file
                              that are part of a file node */
FT_CMD_RETR   = 8,         /* The server requests a file that is part of a
                              file node */
```

15.2.16.3 Callback functions

public func(id, cmd, const data{}, len, ofs);

Function to be provided by the device logic developer, that is called up when a file transfer command is received. The callback function must be able to handle all file transfer commands (see "File transfer commands" in chapter "Constants" on page 212).

<table>
<tr><th>Parameter</th><th>Explanation</th></tr>
<tr><td>id</td><td>Unique identification with which the file is referenced (specified during registration)</td></tr>
<tr><td>cmd</td><td>File transfer command that was received from the system and that has to be processed by the callback function</td></tr>
<tr><td>data</td><td>This parameter is only relevant when the following file transfer commands are received:
FT_CMD_STORE: Array that contains the properties of the file that should be newly created. Structure:<table><tr><th>Offset</th><th>Bytes</th><th>Explanation</th></tr><tr><td>0</td><td>4</td><td>Time stamp of the file</td></tr><tr><td>8</td><td>4</td><td>File size in byte</td></tr><tr><td>12</td><td>4</td><td>Ethernet CRC32 of the file</td></tr><tr><td>16</td><td>2</td><td>File flags</td></tr><tr><td>18</td><td>256</td><td>File name</td></tr></table>
FT_CMD_WRITE: Array that contains the data received from the myDatanet server.
FT_CMD_RETR: Name of a file (ASCII) that is part of a file node and was requested by the server</td></tr>
<tr><td>len</td><td>This parameter is only relevant when the following file transfer commands are received:
FT_CMD_READ: Number of bytes requested by the myDatanet server (max. 4kB)
FT_CMD_STORE: Size of the file property block received from the myDatanet server
FT_CMD_WRITE: Number of bytes received from the myDatanet server
FT_CMD_RETR: Length of the file name (number of characters excluding the final '\0')</td></tr>
<tr><td>ofs</td><td>This parameter is only relevant when the following file transfer commands are received:
FT_CMD_READ: Byte offset within the file of the data block to be transferred to the myDatanet server
FT_CMD_WRITE: Byte offset within the file of the data block received from the myDatanet server
FT_CMD_ENUM: List index (starting with 0) for when the server requests the properties of a file that belongs to a file node[1)]</td></tr>
</table>

[1)]Upon receipt of the file transfer command "FT_CMD_ENUM", the "FT_SetPropsExt()" function can be used to set the properties of a file that should be assigned to the current file node. This means that a file is assigned to the file node. Following the first "FT_CMD_ENUM" command, the system sends further "FT_CMD_ENUM" commands until the device logic developer indicates that they do not want to assign any more files to the current file node. The developer must indicate this by setting the length for the "TFT_Info" structure (i.e. the "len" parameter) to 0 when setting the file properties via the "FT_SetPropsExt()" function.

15.2.16.4 Functions

native FT_Register(const name{}, id, funcidx);

Registers a file made available by the device logic.

Parameter	*Explanation*
name	*Unique file name*
id	*Unique identification with which the file is subsequently referenced (freely selectable)*
funcidx	*Index of the public function that should be called up if a file transfer command has been received* *Type of function: public func(id, cmd, const data{}, len, ofs);* ***Important note:** All file transfer commands (see "File transfer commands" in chapter "Constants" on page 212) must be handled by this public function.*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_RegisterEnum(id, funcidx, props[TFT_Info], len=sizeof props);

Registers a file node made available by the device logic. Several files can be managed via a file node.

Parameter	*Explanation*
id	*Unique identification with which the file node is subsequently referenced (freely selectable)*
funcidx	*Index of the public function that should be called up if a file transfer command has been received* *Type of function: public func(id, cmd, const data{}, len, ofs);* ***Important note:** All file transfer commands (see "File transfer commands" in chapter "Constants" on page 212) must be handled by this public function.*
props	*Structure that contains the properties of a file entry for the file node (see "TFT_ Info" in chapter "Arrays with symbolic indices" on page 212)* *.name: Those parts of the file node name that are not wildcards, must also appear in the names of the files that should be managed with the file node (wildcard matching), so that the read and write operations are accepted.* *.flags: Read/write flags as required; node flag mandatory*
len	*Size (in cells) of the structure that contains the properties of a file entry – OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_Unregister(id);

Removes a file from the registration. The file is no longer available for the file transfer.

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_SetProps(id, stamp, size, crc, flags);

Sets the properties of a file

Important note: *This function must be called up following receipt of a "FT_CMD_LIST" command.*

Important note: *Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects. The "FT_SetPropsExt()" function should be used as an alternative.*

Parameter	***Explanation***
id	*Unique identification with which the file is referenced (specified during registration)*
stamp	*Time stamp of the file (seconds since 31.12.1999)*
size	*File size in byte*
crc	*Ethernet CRC32 of the file*
flags	*File flags (see "File flags" in chapter "Constants" on page 212)*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_SetPropsExt(id, props[TFT_Info], len=sizeof props);

Sets the properties of a file (extended format)

Important note: *This function must be called up following receipt of a "FT_CMD_LIST" command.*

Parameter	***Explanation***
id	*Unique identification with which the file is referenced (specified during registration)*
props	*Structure that contains the properties of a file entry (see "TFT_Info" in chapter "Arrays with symbolic indices" on page 212)*
len	*Size (in cells) of the structure that contains the properties of a file entry – OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_Read(id, const data{}, len);

Transmits the data to the system, to transfer it to the myDatanet server. The data must be provided by the callback function specified via "FT_Register()".

> ***Important note:*** *This function must be called up following receipt of a "FT_CMD_READ" command.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
data	*Array that contains the data that should be transmitted to the system to be transferred to the myDatanet server*
len	*Number of bytes to be transferred*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_Accept(id, newid=-1);

Accepts the file that the myDatanet server wants to write. It is a new file if the transferred unique identification number ("id" parameter) refers to a file node. In this case, an unique identification number ("newid" parameter) must be assigned to the new file. The new file must also be registered via the "FT_Register()" function. The file properties that were transmitted by the system to the callback function (see "Callback functions" on page 212) must be saved via the "FT_SetProps()" function.

> ***Important note:*** *This function must be called up following receipt of a "FT_CMD_STORE" command.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
newid	*Unique identification for the new file that should be created (-1 if it is not a new file) - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_Written(id, len);

Confirms that the data received from the myDatanet server has been written. The actual writing process must be executed via the callback function specified via "FT_Register()". The data that is to be written is transmitted to the callback function by the system (see "Callback functions" on page 212).

> ***Important note:*** *This function must be called up following receipt of a "FT_CMD_WRITE" command.*

Parameter	***Explanation***
id	*Unique identification with which the file is referenced (specified during registration)*
len	*Number of written bytes*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native FT_Error(id);

Used to display a file handling error and terminates any file command.

Parameter	***Explanation***
id	*Unique identification with which the file is referenced (specified during registration)*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

15.2.17 LED

15.2.17.1 Constants

Selection of whether the two-colour LED is controlled by the firmware or script

Configuration options for the Led_Init() function

```
LED_MODE_INTERNAL = 0,          // controlled by the FW
LED_MODE_SCRIPT   = 1,          // controlled by the script
```

15.2.17.2 Functions

native Led_Init(mode);

Initialises the two-colour LED

Parameter	*Explanation*
mode	*Selection of whether the two-colour LED is controlled by the firmware or script* *LED_MODE_INTERNAL :* *The two-colour LED is used to indicate the operating state (see "Three-colour LED" on page 84).* *LED_MODE_SCRIPT :* *The state of the two-colour LED can be controlled by the "Led_On", "Led_Off", "Led_Blink" and "Led_Flash" script functions.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native Led_Close();

Deactivates the two-colour LED. The two-colour LED cannot be controlled by the firmware or the script functions.

	Explanation
Return value	*OK, if successful*

native Led_On(bool:red, bool:green);

The two-colour LED consists of a red and a green LED that can be switched on separately by this function. If both LEDs are switched on simultaneously, the two-colour LED lights up orange.

Parameter	*Explanation*
red	*true: The red LED is switched on.*
green	*true: The green LED is switched on.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native Led_Off(bool:red, bool:green);

The two-colour LED consists of a red and a green LED that can be switched off separately by this function.

Parameter	*Explanation*
red	*true: The red LED is switched off.*
green	*true: The green LED is switched off.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native Led_Blink(red, green);

Enables the two-colour LED to flash (t_{on} = 500ms , t_{off} = 500ms). The two-colour LED consists of a red and a green LED. If both LEDs are used, the two-colour LED flashes orange.

Parameter	*Explanation*
red	*-1 : The red LED remains switched off.* *0 : The red LED flashes until it is deliberately switched off.* *>0 : Number of times the red LED should flash*
green	*-1 : The green LED remains switched off.* *0 : The green LED flashes until it is deliberately switched off.* *>0 : Number of times the green LED should flash*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native Led_Flash(red, green);

Enables the two-colour LED to briefly flash every 500ms . The two-colour LED consists of a red and a green LED. If both LEDs are used, the two-colour LED briefly flashes orange.

Parameter	*Explanation*
red	*-1 : The red LED remains switched off.* *0 : The red LED briefly flashes at regular intervals until it is deliberately switched off* *>0 : Number of times the red LED should briefly flash at regular intervals*
green	*-1 : The green LED remains switched off.* *0 : The green LED briefly flashes at regular intervals until it is deliberately switched off* *>0 : Number of times the green LED should briefly flash at regular intervals*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native Led_Flicker(red, green);

Enables the two-colour LED to flicker (t_{on} = 94ms , t_{off} = 31ms). The two-colour LED consists of a red and a green LED. If both LEDs are used, the two-colour LED flickers orange.

Parameter	*Explanation*
red	*-1 : The red LED remains switched off.* *0 : The red LED flickers until it is deliberately switched off.* *>0 : Number of times the red LED should flicker*
green	*-1 : The green LED remains switched off.* *0 : The green LED flickers until it is deliberately switched off.* *>0 : Number of times the green LED should flicker*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

15.2.18 Solenoid switch

15.2.18.1 Constants

Selection of whether the solenoid switch is evaluated by the firmware or script

Configuration options for the Switch_Init() function

```
SWITCH_MODE_INTERNAL = 0,        // evaluation by the FW
SWITCH_MODE_SCRIPT   = 1,        // evaluation by the script
```

15.2.18.2 Callback Funktionen

public func(key);

Function to be provided by the script developer, that is called when the state of the button changes

Parameter	*Explanation*
key	*Indicates which state change called the function* *0: Button was released* *1: Button was pressed*

public func(key);

Function to be provided by the script developer, that is called when the state of the lid reed contact changes

Parameter	*Explanation*
key	*Indicates which state change called the function* *0: Housing cover has been opened* *1: Housing cover has been closed*

15.2.18.3 Functions

native Switch_Init(mode, funcidx=-1);

Initialises the solenoid switch

Parameter	*Explanation*
mode	*Selection of whether the solenoid switch is evaluated by the firmware or script* *SWITCH_MODE_INTERNAL :* *If the button was pressed and held for 3 sec., a transmission is initiated when the button is released.* *SWITCH_MODE_SCRIPT :* *In the event of a state change of the button (press or release), the public function, for which the index was transferred to the "Switch_Init" function, is called.*
funcidx	*Index of the public function that should be executed in the event of a state change of the button (only necessary if mode=SWITCH_MODE_SCRIPT)* *Type of function: public func(key);*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native Switch_Close();

Deactivates the solenoid switch. An action is no longer initiated when the button is pressed.

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native LidCover_Init(mode, funcidx=-1);

Initialises the lid reed contact

Parameter	*Explanation*
mode	*reserved for extensions; must be set to 0*
funcidx	*Index of the public function that should be executed in the event of a state change of the lid reed contact* *Type of function: public func(key);*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

native LidCover_Close();

Deactivates the lid reed contact. When the housing cover is opened/closed, no action is triggered any more.

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 132)*

15.2.19 Power management

15.2.19.1 Arrays with symbolic indices

TPM_Info

Information on the energy source used and power management status

```
// BatteryType     PSU type (see "Type of power supply unit" in chapter
//                 "Constants" on page 224)
// Flags           Power management status (see "Power management status"
//                 in chapter "Constants" on page 224)
// VIn             Supply or charging voltage V IN in [mV]
// VBatt           Battery or rechargeable battery voltage in [mV]
// SOC             State of charge in [0.01%]
// PIn             Power consumption in [mW]
// ChargingMode    Charge mode (see "Charge mode" in chapter
//                 "Constants" on page 224)
// Description     Designation of the power supply unit

#define TPM_Info[  .BatteryType, .Flags, .VIn, .VBatt, .SOC, .PIn,
                   .ChargingMode, .Description{16}                ]
```

15.2.19.2 Constants

Charging mode

```
PM_CHARGING_OFF    = 0,          // Charge control deactivated
PM_CHARGING_NORMAL = 1,          // charge, if state of charge <50%
PM_CHARGING_SOLAR  = 2,          /* always charge when possible and sufficient
                                    input voltage (V IN > 16V ) is available */
```

Type of power supply unit

```
PM_BATT_TYPE_NONE   = 0,         // No battery or external power supply
PM_BATT_TYPE_NORMAL = 1,         // Battery, only discharged
PM_BATT_TYPE_LIIO   = 2,         // li-ion rechargeable battery
```

Power management status

```
PM_FLAG_CHARGING      = 0x00000001, // Charge control active
PM_FLAG_BACKUP        = 0x00000002, /* Supply via internal energy source
                                       following failure of V IN (only if
                                       V IN monitoring was activated) */
PM_FLAG_ERROR         = 0x00000008, /* Power management error (does not charge,
                                       SoC cannot be determined, because the
                                       memory of the power supply unit could
                                       not be read, for example) */
PM_FLAG_AKKU_FAULT    = 0x00000010, /* The rechargeable battery has been
                                       marked defective and can no longer be
                                       charged. */
```

Configuration of the Coulomb counter (electric charge)

Configuration flags for the PM_GetCoulombCounter() function

```
PM_CC_RESET   = 0x00000001,         /* Reset Coulomb counter that can be used
                                       for the application */
```

15.2.19.3 Callback functions

public func();

Function to be provided by the script developer, that is called up if a failure of the supply or charging voltage V IN has been detected

15.2.19.4 Functions

native PM_SetChargingMode(mode);

Sets the charge mode. Detailed information on the charge control is provided in chapter "Technical details about energy management" on page 75.

Parameter	*Explanation*	
mode	*PM_CHARGING_OFF :*	*Charge control deactivated*
	PM_CHARGING_NORMAL :	*Charge if the state of charge of the rechargeable battery is <50%.*
	PM_CHARGING_SOLAR :	*Charge continuously, if possible, and the supply or charging voltage V IN is above 16V*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred* • *ERROR-1, if the reghargeable battery has been marked as defective and can no longer be charged.*

native PM_BackupInit(funcidx);

Activates malfunction monitoring for the supply or charging voltage V IN and specifies the function that should be called up in the event of the supply or charging voltage V IN failing

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up if a failure of the supply or charging voltage V IN has been detected* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native PM_BackupClose();

Deactivates malfunction monitoring for the supply or charging voltage V IN

	Explanation
Return value	*OK, if successful*

native PM_GetInfo(info[TPM_Info], len=sizeof info);

Provides information on the energy source used and power management status

Parameter	*Explanation*
info	*Structure for storing the information (see "TPM_Info" in chapter "Arrays with symbolic indices" on page 223)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *ERROR, if an invalid parameter was transferred*

native PM_GetCoulombCounter(flags=0);

Reads the current state of the Coulomb counter (electric charge) that can be used for the application derived from the system's internal Coulomb counter

Parameter	*Explanation*
flags	*Configuration flags to be set/deleted - OPTIONAL* *Bit0: Resetting the Coulomb counter that can be used for the application* *0 = no action* *PM_CC_RESET = Counter reset*

	Explanation
Return value	*Depleted electric charge [mAs] since the last reset of the Coulomb counter that can be used for the application*

15.3 Device Logic error codes

If an error occurs while executing the Device Logic, the corresponding error code is entered in the device log and the Device Logic is restarted. If such an error occurs more than three times in 24 hours, the Device Logic is deactivated and error handling is activated (see "Error handling" on page 44). The parameter for all log entries except "SCRIPT_ERR" contains the 32-bit instruction pointer of the Abstract machine (AMX). Two entries are generated in the device log as only 16-bit values can be saved in the parameter of a log entry. The first entry contains Bit31-Bit16 and the second entry contains Bit15-Bit0 of 32-bit instruction pointer. Instructions on evaluating the device log are included in the chapter "Log tab" (see ""Log" tab" on page 101).

Log entry		Parameter		Description
Code	**Plain text**	**Code**	**Plain text**	
3000	SCRIPT_ERR	0	NO SCRIPT	No valid Device Logic available
		1	SCRIPT UPDATE	New Device Logic received
		2	SCRIPT EXCEPT LOOP	Exception loop detected (4 system starts due to exception within 10 min.). The Device Logic is deactivated and error handling is activated (see "Error handling" on page 44). The file system is also reformatted. This means that all of the data and log entries recorded to date are deleted. This is indicated by the additional "LOG REFORMATFILE" log entry.
		3	SCRIPT SOFT ERROR 1	First time a runtime error has occurred within 24 hours. Device Logic was restarted.
		4	SCRIPT SOFT ERROR 2	Second time a runtime error has occurred within 24 hours. Device Logic was restarted.
		5	SCRIPT SOFT ERROR 3	Third time a runtime error has occurred within 24 hours. Device Logic was restarted. The Device Logic is deactivated and error handling is activated if another runtime error should occur within 24 hours (see "Error handling" on page 44).
		6	SCRIPT UPDATE ERROR	Error when installing the device logic received during the download. The "Interval & Wakeup" connection type was activated and a connection to the server is established every hour.
		7	SCRIPT SYSTEM SHUTDOWN	Reserved for extensions
		8	SCRIPT DOWNLOAD ERROR	Connection aborted while downloading the device logic. Although this does not affect the existing Device Logic. It can continue to be executed.
		9	SCRIPT DELETED	The device logic was deleted manually (e.g. using rapidM2M Studio). The "Interval & Wakeup" connection type was activated and a connection to the server is established every 24 hours.
3001	AMX_ERR_EXIT	##	---	Abortion e.g. max. number of commands (100,000) per run reached

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3002	AMX_ERR_ ASSERT	##	---	Assertion failed
3003	AMX_ERR_ STACKERR	##	---	Stack/heap collision (insufficient stack size)
3004	AMX_ERR_ BOUNDS	##	---	Array index outside the valid range
3005	AMX_ERR_ MEMACCESS	##	---	Invalid memory access e.g. mix-up between cell (32-bit element) access [] and byte access {}
3006	AMX_ERR_ INVINSTR	##	---	Invalid statement
3007	AMX_ERR_ STACKLOW	##	---	Stack underflow
3008	AMX_ERR_ HEAPLOW	##	---	Heap underflow
3009	AMX_ERR_ CALLBACK	##	---	No (invalid) native callback function
3010	AMX_ERR_NATIVE	##	---	Native function failed
3011	AMX_ERR_DIVIDE	##	---	Division by zero
3012	AMX_ERR_SLEEP	##	---	Sleep mode
3013	AMX_ERR_ INVSTATE	##	---	Invalid state
3014	reserved			
3015	reserved			
3016	AMX_ERR_ MEMORY	##	---	Out of memory
3017	AMX_ERR_ FORMAT	##	---	P-code file format is invalid/not supported
3018	AMX_ERR_ VERSION	##	---	File is for a newer version of AMX
3019	AMX_ERR_ NOTFOUND	##	---	File or function not found
3020	AMX_ERR_INDEX	##	---	Invalid index parameter (invalid entry point)
3021	AMX_ERR_DEBUG	##	---	Debugger cannot be executed
3022	AMX_ERR_INIT	##	---	AMX not initialised (or initialised twice)
3023	AMX_ERR_ USERDATA	##	---	User data field cannot be set (table full)

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3024	AMX_ERR_INIT_ JIT	##	---	JIT cannot be initialised.
3025	AMX_ERR_ PARAMS	##	---	Faulty parameter
3026	AMX_ERR_ DOMAIN	##	---	Domain error. The result of the expression is not in the valid range.
3027	AMX_ERR_ GENERAL	##	---	General error (invalid or non-specific error)
3028	AMX_ERR_ OVERLAY	##	---	Overlays are not supported (JIT) or are not initialised.
3050	LOG_NOSCRIPT_ ERR	1	SCRIPT File not available	Meta information on the Device Logic not available
		2	SCRIPT File not fully downloaded	Inconsistency between the meta information on the Device Logic and the Device Logic itself detected
		3	SCRIPT Error reading file	Error detecting the Device Logic type
		4	SCRIPT Marked faulty	Device Logic is marked as faulty (4 runtime errors have occurred within 24 hours)
		5	SCRIPT Invalid type	Invalid Device Logic Typ detected
		6	SCRIPT Error loading program	Error initializing the Device Logic
		7	SCRIPT Exception loop detected	Exception Loop detected (4 system starts due to exception within 10 min.)
		8	SCRIPT Invalid filesize	If the size of the Binarys is <= 1 Byte

15.4 Syntax

15.4.1 General syntax

15.4.1.1 Format

Identifiers, numbers and characters are separated by spaces, tabs, line breaks and "form feed". A series of one or more of these separators is recognised as an empty space.

15.4.1.2 Optional semicolons

Semicolons (used to finish a statement) are optional if they are at the end of a line. Semicolons are required to separate several statements in a line. An expression can be split across several lines, though the postfix operators must be on the same line as the operand.

15.4.1.3 Comments

Text between the /* and */ symbols (both symbols can be on the same or different lines) and text following // (to the end of the line) are comments. Comments must not be nested. The compiler considers comments to be blank space. A documentation comment is a comment that starts with "/** " (two stars and space after the second star) and ends with "*/". A comment that starts with "/// " (three forward slashes and a space after the third slash) is also a documentation comment. The parser can support the documentation comment in different ways, for example, by using it to generate online help.

15.4.1.4 Identifier

Names of variables, functions and constants. Identifier comprises the characters a...z, A...Z, 0...9, _ or @. The first character must not be a number. The characters @ and _ on their own are not valid identifiers, e.g. "_Up" is a valid identifier but "_" is not. A distinction is made between upper and lower case. The parser cuts identifiers off after a certain length. By default, only the first 16 characters are referenced for distinguishing purposes.

15.4.1.5 Reserved keywords

Statements	Operator	Directives	Others
assert	defined	defined	defined
break	sizeof	sizeof	sizeof
case	state	state	state
continue	tagof	tagof	tagof
default			
do			
else			
exit			
for			
goto			
if			
return			
sleep			
state			
switch			
while			

15.4.1.6 Numerical constants

15.4.1.6.1 Numerical integer constants

Binary
0b followed by a series of 0 and 1

Decimal
A series of numbers between 0 and 9

Hexadecimal
0x followed by a series of numbers between 0 and 9 and the letters a to f

15.4.1.6.2 Numerical floating-point constants

A floating-point number is a number with numbers after the decimal point. A floating-point number starts with one or several numbers, includes a decimal point and has at least one number after the decimal point, e.g. "12.0" and "0.75" are valid floating-point numbers. An exponent can optionally be added. The notation is the letter "e" (lower case) followed by an integer numerical constant. For example, "3.12e4" or "12.3e-3" are valid floating-point numbers with an exponent.

15.4.2 Variables

15.4.2.1 Declaration

The keyword "new" declares a new variable. For special declarations, the keyword "new" is replaced with "static" (see "Static local declaration" on page 232). The value of the new variable is zero, provided that is not initialised explicitly.

A variable declaration can appear

- At every position at which an expression is valid - local variable
- At every position at which a function declaration or the implementation of the function is valid - global variables
- In the first expression of a "for" loop (see "For (expression 1; expression 2; expression 3) statement" on page 242) - local variable

Example:

```
new a;     // without initialisation (value is 0)
new b = 3; // with initialisation (value is 3)
```

15.4.2.2 Local declaration

A local declaration appears within a statement block. A variable can only be accessed within this block and the blocks that it comprises. A declaration within the first expression of a loop instruction is also a local declaration.

15.4.2.3 Global declaration

A global declaration appears outside of a function and a global variable can be used in any function. Global variables can only be initialised with constant expressions.

15.4.2.4 Static local declaration

A local variable is destroyed if the execution leaves the block in which the variable was created. Local variables in a function only exist during the operating time of the specified function. Each new call up of the function creates and initialises new local variables. The variable will also remain in the memory at the end of the function, if a local variable is declared with the keyword "static" instead of "new". This means that static, local variables provide permanent private storage that can only be accessed by a single function (or block). Static local variables can only be initialised with constant expressions, the same way as global variables.

15.4.2.5 Static global declaration

A static global variable acts in the same way as a global variable with the difference that the variable is only valid in the file in which it was declared. Replace the keyword "new" with "static" to declare a global variable as static.

15.4.2.6 Floating point values

Floating point values are supported. These can be added at every point at which a variable declaration is valid.

Example:

```
new Float:a;       // without initialisation (value is 0.0)
new Float:b = 3.0; // with initialisation (value is 3.0)
```

15.4.3 Constant variables

It is sometimes necessary to create a variable that is initialised once and is then not meant to be changed again. Such a variable acts in a similar way to a symbolic constant although it is still a variable. To declare a constant variable, place the keyword "const" between the keyword that starts the variable declaration ("new", "static") and the name of the variables.

Example:

```
new const address[4] = { 192, 0, 168, 66 }
static const status /* initialised to zero */
```

Typical situations in which you could use a constant variable, include:

- To create an "array" constant. Symbolic constants cannot be accessed via the index.
- It is a special case when array arguments are marked as "const" in a function. Arrays arguments are always transferred via a reference. If the arguments are declared to be "const", they are protected against unwanted changes. See examples of "const function arguments" in the chapter "Function arguments ("call-by-value" versus "call-by-reference")" on page 245.

15.4.4 Array variables

15.4.4.1 One-dimensional arrays

The name[constant] syntax declares the name as an array of "constant" elements, where each element is an entry. "name" is a placeholder for the name of the variable and "constant" is a positive value not equal to zero. "constant" is optional and can be omitted. If there is no value between the brackets, the number of elements is equal to the number of initial values. The array index area is "zero-based", which means that the first element is "name[0]" and the last element is "name[constant-1]".

15.4.4.2 Initialisation

Data objects can be initialised during their declaration. The initialised value from global data objects must be a constant value. Global or local arrays must also be initialised with constant values. Data that is not initialised are zero by default.

Example:

List: valid declaration

```
new i = 1
new j                                   /* j is 0 */
new k = 'a'                             /* k has the character code of 'a' */
new a[] = [1,4,9,16,25]                 /* a has 5 elements */
new s1[20] = ['a','b']                  /* the remaining 18 elements are 0 */
new s2[] = ''Hello world...''           /* an unpacked string */
```

List: invalid declaration

```
new c[3] = 4                            /* An array cannot be set to an individual
                                           value */
new i = "Good-bye"                      /* only an array can hold a string */
new q[]                                 /* Unknown size for an array */
new p[2] = { i + j, k - 3 }             /* Array initialisers must be constants */
```

15.4.4.3 Progressive initialisation for arrays

The point operator continues the initialisation of the arrays based on the last two initialised values. The point operator (three points, "...") initialises the array up to the array limit.

Example: List: array initialisers

```
new a[10] = { 1, ... }                  // sets all of the elements to 1
new b[10] = { 1, 2, ... }               // b = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
new c[8]  = { 1, 2, 40, 50, ... }       // c = 1, 2, 40, 50, 60, 70, 80, 90
new d[10] = { 10, 9, ... }              // d = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

15.4.4.4 Multi-dimensional arrays

(Only arrays with up to three dimensions are supported)

Multi-dimensional arrays are arrays that include references to other sub-arrays. For example, a two-dimensional array is an "array on one-dimensional arrays".

Example for the declaration of two-dimensional arrays:

```
new a[4][3]
new b[3][2] = [ [ 1, 2 ], [ 3, 4 ], [ 5, 6 ] ]
new c[3][3] = [ [ 1 ], [ 2, ...], [ 3, 4, ... ] ]
new d[2]{10} = [ "agreement", "dispute" ]
```

```
new e[2][] = [ ''OK'', ''Cancel'' ]
new f[][] = [ ''OK'', ''Cancel'' ]
```

As the last two declarations (variables "e" and "f") illustrates, the last dimension has an unspecified length. In this case, the length of the sub-array is detected by the associated initialiser. Each sub-array is a different length. In this specific example, "e[1][5]" includes the letter "l" of the word "Cancel". However, "e[0][5]" is invalid as the sub-array e[0] only comprises three entries (the letters "O", "K" and the zero terminator). The difference between the declarations of the "e" and "f" arrays is that we enable the compiler to determine the number of higher dimensions for "f". "sizeof f" and "sizeof e" are both 2 (see "Arrays and the "sizeof" operator" on page 234).

15.4.4.5 Arrays and the "sizeof" operator

The "sizeof" operator returns the number of elements of a variable. The "sizeof" result of a simple (non array) variable is always 1.

An array with one dimension comprises a number of elements and the "sizeof" operator returns this quantity. The code section below would therefore issue "5", as the array comprises four characters and the zero terminator.

```
new msg[] = ''Help''
printf(''%d'', sizeof msg);
```

The "sizeof" operator always returns the number of entries even for a "packed" array. The code section below also issues "5", as the variable comprises five entries even though it requires less memory space.

```
new msg{} = "Help"
printf(''%d'', sizeof msg);
```

For multi-dimensional arrays, the "sizeof" operator can return the number of elements for every dimension. An element in the last (lowest) dimension is a single entry, while it is a sub-array in the highest dimension. Please note that in the following code section, the "sizeof matrix" syntax returns the number of elements of the higher dimension and that the "sizeof matrix[]" syntax issues the lower dimension of the two-dimensional array. The code section issues three (higher dimension) and two (lower dimension).

```
new matrix[3][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } }
printf(''%d %d'', sizeof matrix, sizeof matrix[]);
```

The application of the "sizeof" operator on multi-dimensional arrays is particularly practical when it is used as a standard value for function arguments.

15.4.5 Operators and expressions

15.4.5.1 Notational conventions

The use of some operators is dependent on the relevant type of operand. The following notations are therefore used in this chapter:

e

Any expression

v

Any expression that can be assigned a value ("lvalue" expression - variable)

a

An array

f

A function

s

A symbol - this can be a variable, a constant or a function

15.4.5.2 Expressions

An expression consists of one or several operands with an operator. The operand can be a variable, a constant or another expression. An expression followed by a semicolon is a statement.

Examples of expressions:

```
v++ f(a1, a2)
v = (ia1 * ia2) / ia3
```

15.4.5.3 Arithmetic

Operator	Example	Explanation
+	e1 + e2	Result of adding e1 and e2
-	e1 - e2	Result of subtracting e2 from e1
	-e	Result of the arithmetic negation of e (two's complement)
*	e1 * e2	Result of multiplying e1 with e2
/	e1 / e2	Result of dividing e1 by e2. The result is truncated to the closest whole number that is less or equal to the quotient. Positive and negative values are rounded down (negative infinity).
%	e1 % e2	Result is the remainder of the division of e1 by e2. The prefix is the same as that of e2
++	v++	Increases v by 1. The result of the expression is the value before the increase.
	++v	Increases v by 1. The result of the expression is the value following the increase.
--	v--	Decreases v by 1. The result of the expression is the value before the decrease.
	--v	Decreases v by 1. The result of the expression is the value following the decrease.

Note: *The unary + is not defined. The operators ++ and -- change the operand. The operand must be a "lvalue".*

15.4.5.4 Bit manipulation

Operator	Example	Explanation
~	~e	The result is the one's complement of e.
>>	e1 >> e2	The result of the arithmetic shift to the right of e1 by e2 bits. The shift is signed: The bit on the far left is copied to the free bits of the result.
>>>	e1 >>> e2	The result of the logical shift to the right of e1 by e2 bits. The shift is unsigned. The free bits of the result are filled with 0.
<<	e1 << e2	Result: Shift to the left of e1 by e2 bits. The free bits of the result are filled with 0. There is no difference between an arithmetic and a logical shift to the left.
&	e1 & e2	The result is the bitwise logical "and" of e1 and e2.
|	e1 | e2	The result is the bitwise logical "or" of e1 and e2.
^	e1 ^ e2	The result is the bitwise logical "exclusive or" of e1 and e2.

15.4.5.5 Assignment

The result of an assignment expression is the value of the operand following the assignment.

Operator	Example	Explanation
=	v = e	Assigns the value of e to the variable v
	v = a	Assigns the array a to variable v. v must be an array of the same size and with the same dimensions as a. a can be a character string or an array.

Note: *The following operators combine an assignment with an arithmetic or bitwise operation. The result of the expression is the value of the left operand following the arithmetic or bitwise operation.*

Operator	Example	Explanation
+=	v += e	Increases v by e
-=	v -= e	Decreases v by e
*=	v *= e	Multiplies v with e
/=	v /= e	Divides v by e
%=	v %= e	Assigns v the remainder of the division of v and e
>>=	v >>= e	Arithmetically shifts v to the right by e bits
>>>=	v >>>= e	Logically shifts v to the right by e bits
<<=	v <<= e	Shifts v to the left by e bits
&=	v &= e	Executes a bitwise "and" from v and e and assigns the result to v
|=	v |= e	Executes a bitwise "or" from v and e and assigns the result to v
^=	v ^= e	Executes a bitwise "exclusive or" from v and e and assigns the result to v

15.4.5.6 Comparative operators

A logical "false" is represented by an integer value of 0; a logical "true" is represented by a value that is not 0. Results of a comparative expression are either 0 or 1 and the "tag" is set to "bool".

Operator	Example	Explanation
==	e1 == e2	The result is "true" if e1 and e2 are the same.
!=	e1 != e2	The result is "true" if e1 and e2 are not the same.

Note: *The following operators can be linked, the same as in the expression "e1 <= e2 <= e3". This means that the result is "1" if every single comparison is true and "0" if at least one comparison is false.*

Operator	Example	Explanation
<	e1 < e2	The result is a logical "true" if e1 is less than e2.
<=	e1 <= e2	The result is a logical "true" if e1 is less or equal to e2.
>	e1 > e2	The result is a logical "true" if e1 is greater than e2.
>=	e1 >= e2	The result is a logical "true" if e1 is greater or equal to e2.

15.4.5.7 Boolean

A logical "false" is represented by an integer value of 0; a logical "true" is represented by a value that is not 0. Results of a comparative expression are either 0 or 1 and the "tag" is set to "bool".

Operator	Example	Explanation
!	!e	The result is a logical "true", if e is logical "false".
||	e1 || e2	The result is "true", if either e1 or e2 (or both) are logical "true". The expression e2 is only evaluated if e1 is logical "false".
&&	e1 && e2	The result is "true" if e1 and e2 are logical "true". The expression e2 is only evaluated if e1 is logical "true".

15.4.5.8 Other

Operator	Example	Explanation
[ ]	a[e]	Array index: The result is the entry at position e of array a.
{ }	a{e}	Array index: The result is the index at position e of "packed" array a:
( )	f(e1, e2, ... eN)	The result is the value that is returned by function f. The function is called up with parameters e1, e2, ... eN. The sequence of the evaluation of the parameters is not defined. (The implementation of the script engine may evaluate the parameters in reverse order.)
? :	e1 ? e2 : e3	The result is either e2 or e3, depending on the value of e1. The conditional expression is a composite expression with a two-part operator, "?" and ":". The expression e2 is evaluated if e1 is logical "true"; e3 is evaluated if e1 is logical "false".
:	tagname: e	"Tag" overwritten: The value of the expression does not change, although the "tag" does change.
defined	defined s	Result is "1" if the symbol was defined. The symbol can be a constant or a global or local variable. The "tag" of the expression is "bool"
sizeof	sizeof s	The result is the number of elements of the specified variable. An element is an entry for simple variables and for one dimensional arrays. For multi-dimensional arrays, the result is the number of elements (sub-arrays) in the highest dimension. Add [] to the name of the array to specify a lower dimension. The result is 0 if the size of the variable is not known. If this operator is used in a "default" value of a function, the expression is executed at the time that the function was called up and not at the time that the definition was completed.
tagof	tagof s	The result is a unique number that represents the "tag" of the variables, the constants, the return value of a function or the name of the "tag" title. If this operator is used in a "default" value of a function, the expression is executed at the time that the function was called up and not at the time that the definition was completed.

15.4.5.9 Priority of the operators

The following table groups the operators with the same priority, starting with the highest priority at the top of the table.

If the evaluation of an expression is not explicitly justified with brackets, it is categorised by the association rules. For example: a*b/c is equal to (a*b)/c based on the left to right association, and a=b=c is equal to a=(b=c).

Operator	Explanation	Reading order
( ) [ ] { }	Function call array index (element) array index (character)	left-to-right
! ~ - ++ -- : defined sizeof tagof	logical not one's complement two's complement (unary minus) increase decrease "tag" overwritten symbol definition status symbol size in "elements" unique number of the tag	right-to-left
* / %	multiplication division modulo	left-to-right
+ -	addition subtraction	left-to-right
>> >>> <<	arithmetic shift to the right logical shift to the right shift to the left	left-to-right
&	bitwise "and"	left-to-right
^	bitwise "exclusive or"	left-to-right
|	bitwise "or"	left-to-right
< <= > >=	less than less or equal to greater than greater or equal to	left-to-right
== !=	equal unequal	left-to-right
&&	logical "and"	left-to-right
||	logical "or"	left-to-right
? :	conditional execution	right-to-left
=	Assignment *= /= %= += -= >>= >>>= <<= &= ^= |=	right-to-left
,	comma	left-to-right

15.4.6 Statements

A statement can comprise one or several lines. A line can comprises two or more statements.

Statements for the sequence control (if, if-else, for, while, do-while and switch) can be nested.

15.4.6.1 Statement label

A label consists of an identifier followed by a ":". A label is a "Jump target" of a "goto" statement.

Each statement can be marked with a label. The label must be followed by a statement, which can also be an "empty statement".

The scope of a label is the function in which it was declared, i.e. a "goto" statement cannot jump from the current function to another function.

15.4.6.2 Composite statements

A composite statement (also known as a block) is a series of zero or several statements that is enclosed by brackets ("{" and "}"). The closing bracket ("}") must not be followed by a semicolon. Each statement can be replaced by a block. A composite statement that does not comprise any statements is a special case and is known as an "empty statement".

15.4.6.3 Expression statement

Each expression becomes a statement when a semicolon (";") is added. An expression also becomes a statement, if the expression is only followed by blank spaces to the end of the line, and the expression is not continued in the next line.

15.4.6.4 Empty statement

An empty statement does not execute any statements and consists of a block statement without statements, i.e. it consists of the "{ }" symbol. Empty statements are implemented in control flow statements without actions (e.g. "while (!iskey()) {}") or if a label is defined exactly before the closing bracket of a block statement. An empty statement does not end with a semicolon.

15.4.6.5 Assert expression

The program is aborted with a runtime error if the expression is logical "false"

> ***Note:*** *This expression protects against "impossible" or invalid conditions. In the following example, a negative fibonacci number is invalid. The assert statement marks this error as a programming error. Assert statements should only ever highlight programmer errors and never user inputs.*

Example:

```
fibonacci(n)
{
  assert n > 0

  new a = 0, b = 1
  for (new i = 2; i < n; i++)
  {
    new c = a + b
    a = b
    b = c
  }
  return a + b
}
```

15.4.6.6 Break

Terminates and leaves the smallest, encircling "do", "for" or "while" statement at any point in the loop. The "break" statement moves the program flow to the next statement outside the loop.

Example:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      break

    a += 1
  }
  return a
}
```

15.4.6.7 Continue

Terminates the current iteration of the smallest encircling "do", "for" or "while" statement and moves the program control to the conditional part of the loop.

Example

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      continue

    a += 1
  }
  return a
}
```

15.4.6.8 Do statement while (expression)

Executes a statement before the conditional part (the "while" condition) is evaluated. The statement is repeated as long as the condition is logical "true". The statement is executed at least once.

Example:

```
example(n)
{
  new a = 0

  do
  {
    a++
  }
  while(n >= 0)

  return a
}
```

15.4.6.9 Exit expression

Cancels the program. The expression is optional, however, if present it must start and end on the same line as the "exit" statement. The exit statement returns the expression value or zero to the main application, if no expression is specified.

15.4.6.10 For (expression 1; expression 2; expression 3) statement

All three of the expressions are optional.

Expression 1:

Is only evaluated once before entering the loop. This expression can be used to initiate a variable. This expression also includes the variable declaration by means of the "new" syntax. A variable that is declared at this stage is only valid in the loop. It is not possible to combine an expression (with existing variables) and a declaration of new variables in this field. All of the variables must either already exist in this field, or they must all be declared in this area.

Expression 2:

This expression is executed before every run of the loop and terminates the loop if the expression logical "false" is returned. If this expression is omitted, it is assumed that the result of expression 2 is logical "true".

Expression 3:

This expression is executed each time the statement is completed. The program control moves from expression 3 to expression 2 for the next (conditional) iteration of the loop.

Example:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n; i++)
  {
    a++
  }

  return a
}
```

The "for (; ;)" statement is the same as the "while (true)" statement.

15.4.6.11 Goto label

Moves the program control (unconditionally) to the statement that follows the specified label. The label must be within the same function as the "goto"-statement (a "goto"-statement cannot jump out of a function).

15.4.6.12 If (expression) statement 1 else statement 2

Executes statement 1 if the results of the expression is logical "true". The "else"clause of the "if" statement is optional. If the result of the expression is logical "false" and there is an "else" clause, the statement that is associated with the "else" clause (statement 2) is executed.

Example:

```
example(n)
{
  if(n < 0)
    return -1
  else if (n == 0)
    return 0
  else
    return 1
}
```

15.4.6.13 Return expression

Terminates the current function and moves the program control to the next statement following the function call. The expression value is returned as the function result. The expression can be an array or a character string. The expression is optional, however, if present it must start on the same line as the "return" statement. Zero is returned if no expression is specified.

15.4.6.14 switch (expression) {case list}

Transfers the sequence control to the various statements within the "switch", depending on the value of the "switch" expression. The main part of the "switch" statement is a composite statement that comprises a series of "case" clauses. Each "case" clause starts with the keyword "case" followed by a list of constants and a statement. The list of constants is a series of expressions separated by commas, each of which is evaluated as a constant value. This list ends with a colon. To specify an area in this list, separate the lower and upper limit of the area with a double point (".."). An example for an area is: "case 1..9:".

The "switch" statement shifts the sequence control to a "case" clause if a value from the list corresponds to the value of the "switch" expression.

The "default" clause consists of the "default" keyword and a double point. The "default" clause is optional, however, if it is specified it must be included as the last entry in the "case" list. The "switch" statement shifts the sequence control to the "default" clause if none of the "case" clauses comply with the "switch" expression.

Example:

```
example(n)
{
  new a = 0

  switch (n)
  {
    case 0..3:
      a = 0
    case 4,6,8,10:
      a = 1
    case 5,7:
      a = 2
    case 9:
      a = 3
    default:
      a = -1
  }

  return a
}
```

15.4.6.15 While (expression) statement

Evaluates the expression and executes the statement if the result of the expression is logical "true". The program control returns to the expression again once the statement has been executed. The statement is therefore executed as long as the expression is logical "true".

Example:

```
example(n)
{
  new a = 0

  while(n >= 0)
  {
    a++
  }

  return a
}
```

15.4.7 Functions

A function declaration specifies the name of the function and the formal parameters enclosed in brackets. A function can also return a value. A function must be defined globally, i.e. declared outside of another function and is globally available.

If the function declaration is followed by a semicolon (instead of a statement), this is a forward declaration of a function.

The "return" statement sets the return value of the function. For example, the return value of the "sum" function (see below) is the sum of both parameters. The "return" expression is optional.

```
sum(a, b)
{
  return a + b
}
```

The arguments of a function are (declared implicitly) local variables for this function. The function call specifies the values of the arguments. Another example of a complete definition of a function is "leap year" that indicates "true" or "false" for the relevant year.

```
leapyear(y)
{
  return y % 4 == 0 && y % 100 != 0 || y % 400 == 0
}
```

Details of the statements used in this example are provided in the chapter "Operators and expressions" on page 235.

Generally, functions include local variable declarations and consist of a block statement.

Note: *In the next example, the "assert" statement prevents negative values for the exponent.*

```
power(x, y)
{
  /* returns x^y */
  assert y >= 0

  new r = 1
  for (new i = 0; i < y; i++)
    r *= x

  return r
}
```

A function can comprise several "return" statements, for example, one is used to quickly terminate a function if invalid parameters are transferred, or when it becomes apparent that the function has nothing to do. If a function returns an array, all of the "return" statements must return an array with the same number of entries.

15.4.7.1 Function arguments ("call-by-value" versus "call-by-reference")

The "faulty" function in the next example has a parameter that is used in the loop to calculate the factorial of this number. It must be noted that the function modifies the argument.

```
main()
{
  new v = 5
  new f = faculty(v)
}

faculty(n)
{
  assert n >= 0

  new result = 1
  while (n > 0)
    result *= n--

  return result
}
```

Regardless of what (positive) value the "n" variable has at the start of the "while" loop, "n" will equal zero at the end of the function. In the "faculty" function, for example, the parameter is transferred as a value ("by value"), which means that changes to the "n" variable are only valid locally in the "faculty" function. In other words, the "v" variable in the "main()" function has the same value before and after the function is called up.

Arguments can be transferred as a value ("by value") or as a reference ("by reference"). A function argument that is to be transferred as a reference must have the "&" prefix preceding the name. The arguments are transferred to the function as a value by default.

Example:

```
swap(&a, &b)
{
  new temp = b
  b = a
  a = temp
}
```

To transfer an array to a function, add a pair of brackets ("[]") to the name of the argument. The number of entries can also be specified. This improves the error detection of the compiler's parser.

Example:

```
addvector(a[], const b[], size)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

Arrays are always transferred as a reference.

> ***Note:*** *The "b" array in the above-mentioned example is not changed in the function. This function argument was declared as a "const" to make this explicit. In addition to the improved error detection, it also enables the compiler to generate a more efficient code.*

The following code example calls up the "addvector" function and adds five to each element of the "vect" variables:

```
new vect[3] = [ 1, 2, 3 ]

addvector(vect, [5, 5, 5], 3)

/* vect[] now comprises the values 6, 7 and 8 */
```

15.4.7.2 Named parameters versus fixed parameters

In the previous examples, the order of the parameters in a function call were important as each parameter was copied to the same position of the function parameter. For example, in the "weekday" function (defined below), the expression "weekday(12 ,31, 1999)" would be used to get the weekday of the last day of the last century.

```
weekday(month, day, year)
{
  /* returns the day of the week: 0=Saturday, 1=Sunday, etc. */
  if (month <= 2)
    month += 12, --year

  new j = year % 100
  new e = year / 100
  return (day + (month+1)*26/10 + j + j/4 + e/4 - 2*e) % 7
}
```

The date format changes depending on the culture and country, while the USA use the month/day/year format, European countries frequently use the day/month/year format and technical publications use the year/month/day (ISO/IEC 8824) format. In other words, the sequence of the parameters is not "standardised" or "normal". For this reason, there is an alternative way of transferring parameters to a function, by using "named parameters". These are illustrated in the next example (the function was declared in the same way as the previous example).

```
new wkday1 = weekday( .month = 12, .day = 31, .year = 1999)
new wkday2 = weekday( .day = 31, .month = 12, .year = 1999)
new wkday3 = weekday( .year = 1999, .month = 12, .day = 31)
```

In "named parameters", a dot (".") precedes the name of the argument. The argument of the function can be set to any expression that is valid for the argument. In the event of a named parameter, the equals sign ("=") does not refer to an allocation but instead links the expression with a function argument.

Fixed and named parameters can be mixed together, although the fixed parameters must be specified before the named parameters.

15.4.7.3 Standard values of function arguments

A function argument can have a standard value. The standard value of a function argument must be a constant. To specify a standard value, add an equals sign ("=") and the value to the name of the parameter.

The standard value is adopted if a placeholder is specified instead of a valid function parameter during a function call. The placeholder is the underscore character ("_"). The argument placeholder is only valid for parameters with a standard value.

The right argument placeholders can be removed from the list of arguments.

For example, if the "increment" function is defined as follows:

```
increment(&value, incr=1)
{
  value += incr
}
```

The following function calls are all the same:

```
increment(a)
increment(a, _)
increment(a, 1)
```

Standard values for arguments that are transferred as a reference are helpful in making these parameters optional. For example, if the "divmod" function was written to return the quotient and the rest as a parameter.

```
divmod(a, b, &quotient=0, &remainder=0)
{
  quotient = a / b
  remainder = a % b
}
```

Based on the previous definition of the "divmod" function, the following function calls are all valid:

```
new p, q

divmod(10, 3, p, q)
divmod(10, 3, p, _)
divmod(10, 3, _, q)
divmod(10, 3, p)
divmod 10, 3, p, q
```

The next example adds the value of an array to another one. The values of the array are increased by one if only one parameter is specified:

```
addvector(a[], const b[] = {1, 1, 1}, size = 3)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

15.5 Differences to C

- The input mechanism of C is not present. It is an "integer-only" variant of C. There are no structures or unions. Floating point support must be implemented with user-defined operators and the help of native functions.
- The syntax for floating point values is stricter than that in C. In contrast to C, values such as ".5" and "6." are not accepted. It is mandatory to write "0.5" and "6.0". The decimal point is optional in C. If an exponent is included, then you can write "2E8" in C. The capital letter "E" is not accepted. Use the lower case letter "e". In addition, it requires a comma: e.g. "2.0e8" (see "Numerical constants" on page 231).

- "pointers" are not supported. A "reference" argument to transfer function parameters as a reference (see "Function arguments ("call-by-value" versus "call-by-reference")" on page 245) is included.. The "placeholder" argument replaces some applications of the ZERO pointer (see "Standard values of function arguments" on page 247).
- Numbers can be specified in a hexadecimal, decimal or binary format. The octal format is not supported (see "Numerical constants" on page 231). Hexadecimal numbers must start with "0x" ("x" in lower case). The prefix "0X" is invalid.
- "Cases" in a "switch"-statement are not "fall through". At least one statement must follow the "case" label. You must create a composite statement (with {}) to execute several statements (see "switch (expression) {case list}" on page 243). The "switch" statement should be considered as a structured "if". However, in C/C++ the "switch" statement is a "conditional goto".
- A "break" statement only terminates loops. In C/C++, the "break" statement also terminates a "case" in a "switch" statement.
- "array assignments" are supported with the limitation that both of the arrays must be the same length. For example, if "a" and "b" arrays have six lines, the expression "a=b" is valid. In addition to character strings, literal arrays and thus expressions such as "a = {0,1,2,3,4,5}" are also supported where "a" is an array variable with six elements.
- "defined" is an operator and not a preprocessor directive. The "defined" operator works with constants (declared with "const"), global variables, local variables and functions.
- The "sizeof" operator returns the size of the variables in "elements" and not in "bytes". An element is an entry or sub-array. Further details are provided in the chapter "Other" on page 238.
- An empty statement is an empty block (with {}) and not a semicolon (see "Composite statements" on page 240). This change prevents frequent errors.
- A division is completed in such a way that the remainder of the division has (or ought to have) the same prefix as the denominator. Divisions (operator "/") are always rounded down to the smaller whole number (whereby -2 is smaller than -1). For example, 5/2 = 2 (2.5 is rounded down to 2), -5/2 = -3 (-2.5 is rounded down to -3). The "%" operator always generates a positive result regardless of the prefix of the numerator (see "Operators and expressions" on page 235).
- There is no unary "+" operator as it is a "no-operation" operator anyway ("a = +1" is not valid; correct: "a = 1").
- Three bit by bit operators have different priorities than in C. The priority level of the "&", "^" and "|" operator is higher than the relational operators. Dennis Ritchie explains that these operators were assigned a low priority level in C as early C compilers did not yet include the logical "&&" and "| |" operators so that bit by bit "&" and "|" were used instead.
- The keyword "const" implements the "enum" functionality of C.

- In most cases, the forward declarations of functions (i.e. prototypes) are not necessary as a two-pass compiler is used. It detects all of the functions during the first cycle and uses them during the second. User-defined operators must however be declared before use. If available, forward declarations must be exactly the same as the definition of the function. The parameter names in the prototypes and the definitions of the functions must be identical. Due to the "named parameter" function, the parameter names in the prototype are of significance. Prototypes are used to call up the forward declared functions. To use these with the named parameters during this process, the compiler must already know the names of the parameters (and their position in the parameter list). The parameter names in the prototypes must therefore match those in the definitions.
- Variables are automatically initialised using „0“. It is therefore not necessary to explicitly set them to „0“.

Chapter 16 Data Descriptor

The basic principle of the myDatalogEASY IoT GW ATEX is "storage-2-storage" data transmission. For this type of data transmission, neither the myDatalogEASY IoT GW ATEX nor the server must know about the logical content of the data blocks. Therefore, the only aim is to transport a block of data from A to B.

The data transferred from the myDatalogEASY IoT GW ATEX to the sever can therefore be selected freely. There are 1023 Byte available per data record that can be used as required. There are also another 10 independent memory blocks each with 4000 Bytes for the configuration data that can be used as required.

The content of the data block or configuration block must be described on the server so that the data and configurations received from the myDatalogEASY IoT GW ATEX can also be used within the myDatanet interface (reports, visualisations, graphics, etc.). The Data Descriptor contains the tool for describing the data as well as the correct provision of the data for use within the interface.

16.1 Data structure

The following containers are available for the different types of data (measurement data, configurations, etc.):

- **Measurement data:** "#histdata0" - "#histdata9"
- **Configurations:** "#config0" - "#config9"
- **Configurations only available on the server:** "#configA" - "#configC"
- **Alarm messages:** #alerts
- **Aloha data:** #aloha

This section describes how the structured measurement data channels ("#histdata0" - "#histdata9"), configuration memory blocks ("#config0" - "#config9") and the aloha data ("aloha") are split into individual data fields for use on the myDatanet server. The structure of the alarm messages ("#alerts") is fixed firmly in the system and does not need to be specified/cannot be changed by the user.

> ***Important note:*** *If a structured measurement data channel, a configuration block or the aloha data are to be available on the myDatanet server or via the REST API then all data fields need to be defined by means of the Data Descriptor .*

An extended example, in which most of the available attributes are used, is provided in chapter "Example" on page 259.

16.1.1 Division of a structured measurement data channel into individual data fields

```
#histdata0 Measurements up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"    vscale=0.001
InputVoltage    s16  title="USB Voltage"      decpl=2  units="V"    vscale=0.001
```

The first line in the example above specifies the container to be used for the measurement data:

#histdata0:	The measurement data should be stored in histdata channel 0.
Measurements:	"Measurements" should be used as the name for the histdata channel.
up:	The data is only transmitted from the device to the server.

Note: *After specifying the direction of transmission other attributes (e.g. "title") could be added.*

The second line in the example above describes the first measurement value in the measurement data container used:

BatVoltage:	"BatVoltage" should be used as the name for the measurement value.
s16:	The data type used for the measurement value should be a 16-bit signed integer.
title:	Name of the measurement value that is displayed on the server
decpl:	Number of decimal places that should be displayed
units:	Unit of the measurement value that is displayed on the server
vscale:	Virtual scaling of the value (see "Attributes of the field definition" on page 254)

Note: *Name and data type must always be specified. Attributes are optional. Further attributes can also be added.*

The third line in the example above describes the second measurement value in the measurement data container used.

16.1.2 Division of a configuration memory block into individual data fields

```
#config0 BasicCfg down title="Basic configuration"
RecordItv        u32  title="Record Interval"        units="sec"  min=10  default=10
TransmissionItv  u32  title="Transmission Interval"  units="min"  min=10  default=60
```

The first line in the example above specifies the container to be used for the configuration:

#config0:	The parameters should be stored in configuration memory block 0.
BasicCfg:	"BasicCfg" should be used as the name for the configuration memory block.
down:	The configuration memory block is only transmitted from the server to the device.
title:	Name of the configuration section that is displayed on the server

Note: *Name and direction of transmission must always be specified. Attributes are optional. Further attributes (e.g. "edit" or "view") can also be added.*

The second line in the example above describes the first parameter in the configuration memory block:

RecordItv:	"RecordItv" should be used as the name for the parameter
u32:	The data type used for the parameter should be a 32-bit signed integer.
title:	Name of the parameter that is displayed on the server
units:	Unit of the parameter that is displayed on the server
min:	smallest valid value for the parameter
default:	Default value for the parameter

Note: *Name and data type must always be specified. Attributes are optional. Further attributes (e.g. "vscale") can also be added.*

The third line in the example above describes the second parameter in the configuration memory block.

16.1.3 Division of the aloha data into individual data fields

```
#aloha up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"    vscale=0.001
InputVoltage    s16  title="USB Voltage"      decpl=2  units="V"    vscale=0.001
```

The first line in the example above specifies the container to be used for the aloha data:

#aloha:	The measurement data should be stored in the aloha data container.
up:	The data is only transmitted from the device to the server.

Note: *After specifying the direction of transmission other attributes (e.g. "title") could be added.*

The second line in the example above describes the first measurement value in the aloha data container used:

BatVoltage:	"BatVoltage" should be used as the name for the measurement value.
s16:	The data type used for the measurement value should be a 16-bit signed integer.
title:	Name of the measurement value that is displayed on the server
decpl:	Number of decimal places that should be displayed
units:	Unit of the measurement value that is displayed on the server
vscale:	Virtual scaling of the value (see "Attributes of the field definition" on page 254).

Note: *Name and data type must always be specified. Attributes are optional. Further attributes can also be added.*

The third line in the example above describes the second measurement value in the aloha data container used.

16.1.4 Attributes of the field definition

title

Alpha-numeric. Title of the field. This title is then used in all of the reports, graphics, etc. for this channel. The length of the title should not exceed 16 characters if possible, otherwise this can cause display problems on the interface.

units

Alpha-numeric. Units of a value.
The length of the units should not exceed eight characters if possible, otherwise this can cause display problems on the interface.

bitmask

Hexadecimal, without leading "0x". Bit mask to mask the actual bits to be used out of the data block, hexadecimal. The extracted value is aligned to the LSB following the masking process.
Example: An u16 field is generated out of two bytes with the 0xF3A7 content. 0FF0 is specified as the bit mask. The bits are extracted and aligned with the LSB. The subsequent HEX value is therefore 0x3A (=58 decimal).

editmask

Format statements for displaying the field content on the interface of the myDatanet server or input via the interface of the myDatanet server.

- *usable for strings (data type "astr", "nstr", "wstr" and "ustr", however not for "cstr")*

Format statements	***Explanation***
"%COLORPICKER%"	*Creates a button that displays the currently selected colour. When clicking the button, a dialog field to set the desired colour appears. The dialog field returns the selected colour as a string in "RRGGBB" format, with the colour components being specified in hex.*
"%HEX%"	*Creates an input field in which the string is displayed in hex format. Each byte of the string is therefore represented by two characters. For example, the letter 'a' is represented as "61".*
"%MASKED%"	*Creates an input field whose content is masked with "asterisk" (e.g. for passwords and tokens). A button (eye icon) for switching between plain text and masking is displayed next to the input field.*
"%MULTILINE%30%75"	*Creates a text field with 30 lines and 75 characters per line*

- *usable for numeric fields (data type "u8", "s8", "f32", ...)*

Format statements	***Explanation***
"%2.x0"	*Creates an input field in which the numeric value is displayed in hex format. Each byte of the data type is therefore represented by two characters. The number after "%" must match the number of characters required to represent the data type (e.g. '4' for a "u16").* ***Note:*** *When using this format statement, the "decpl" attribute must also be set to "-1" (i.e. decpl=-1).*
"0=off;1=on"	*A dropdown list is created in which the text following the "=" is displayed for each entry instead of the value. Entries must be separated using a ";".* ***Note:*** *The entries MUST NOT strart with '+' or '~'*
"%CHECKBOX%"	*A checkbox is created.* *1 = tick set* *not equal to 1 = tick not set* ***Note:*** *If the checkbox is used to display data and the device returns a value not equal to 1, the tick is also displayed as "not set". If the checkbox is used for data entry and the tick is not set, the checkbox returns 0.*
"%CHECKBOX%5;10"	*A checkbox is created the same as for editmask="%CHECKBOX%", accept that the values for "tick set" (in this example 10) and "tick not set" (in this example 5 or not equal to 10) can be chosen. The note above is also valid in a similar way.*
"%TIME%s%hh:mm:ss"	*Creates an input field in which the numeric value is displayed in the specified time format (e.g. hh:mm:ss)* *After the segment "%TIME" it must be specified whether the value is saved in seconds (%s) or in minutes (%m). The time format follows after this specification. "%hh:mm:ss", %hh:mm" or "%mm:ss" are possible formats. Caution "%mm:ss" is not valid if the value is to be saved in minutes (%m).* ***Note:*** *It is advisable to use the "units" attribute to specify the time format (e.g. hh:mm:ss) in which the value is displayed.*

- *usable for timestamps (data type "stamp32" and "stamp40")*

Format statements	Explanation
%DATETIMEPICKER%	Creates an input field in which the timestamp is displayed as date/time. When clicking on the input field, a dialog field to select date and time appears.

decpl

Numeral, integer positive. Number of decimal places that should be displayed.

vscale

Numeral, floating point. Virtual scaling of the value. The extracted value is multiplied by this factor and only then can it continue to be used.
*This value represents value k from the formula k*x+d.*

vofs

Numeral, floating point. Virtual offset of the value. The extracted value is multiplied by vscale, and vofs is then added to the value.
*This value represents value d from the formula k*x+d.*

min

The minimum value for the subsequent display on the server (e.g. graphic).

max

The maximum value for the subsequent display on the server (e.g. graphic). This value is used for the length of the character string in the string (or char) data type. This means that the specification of the max. value for the string (or char) is mandatory.

chmode

The channel mode. This selected setting affects any further processing and display of the channel in the individual server modules.
The following channel modes are available:

Mode	Name	Explanation
1	Digital	The channel is processed as a digital channel. To prevent any problems from occurring, the included value should be 0 or 1.
2	Day counter	A counter for which the value is reset once daily. This reset must be completed by the control program.
3	Interval meter	A meter that is reset every time a measurement data record is created. This reset must be completed by the control program.
6 (standard)	Analogue	A "simple" measurement value, e.g. the temperature
12	Infinite meter	A meter for which the value is never reset, e.g. water or electricity meter

index

Is used for the user-defined sorting of the fields in the selection lists. The standard value for the 1000 channels that are available is -1, which ensures that the channel is hidden.
As soon as a channel is used in the data structure description (a simple field tag with the attribute name will suffice), the background index value is automatically set to the field index (e.g. index=2 for ch2).
This standard sorting can be overridden by specifying the index field.

view

Numeral, integer positive. Specifies from which user level, the field is visible on the interface of the myDatanet server.

edit

Numeral, integer positive. Specifies the user level that is required to be able to change the field content via the interface of the myDatanet server. If this attribute is not specified or if the specified value is lower than that of the "view" attribute, the user level specified for the "view" attribute is required to change the field content.
If changing the field content via the REST-API should be prevented, the value for this attribute must be set to 99.

16.2 Example

```
#histdata0 Measurements up
Delay  u16  title="Delay"           units="sec" min=10  max=2000 vofs=10      chmode=3  index=1
Height f32  title="Height"  decpl=2  units="cm"  min=0   max=2000 vscale=0.01  chmode=6  index=0
Pump   u8   view=99  edit=99
@Pump
Pump_MSK  u8  dlorw=skip  title="Pump"  bitmask=$01  min=0  max=1              chmode=1
Info      astr.50         title="Info"                                                  index=10
```

The first line specifies the container to be used for the measurement data:

#histdata0:	The measurement data should be stored in histdata channel 0.
Measurements:	"Measurements" should be used as the name for the histdata channel
up:	The data is only transmitted from the device to the server

The second line describes the first measurement value "Delay" in the measurement data container used:

Delay:	"Delay" should be used as the name for the measurement value.
u16:	The data type used for the measurement value should be a 16-bit unsigned integer (i.e. 2 bytes).
title:	Name of the measurement value that is displayed on the server
units:	Unit of the measurement value that is displayed on the server
min:	Minimum value for the further display on the server (e.g. graphic)
max:	Maximum value for the further display on the server (e.g. graphic)
vofs:	Virtual offset of the value (see "Attributes of the field definition" on page 254). In the current example 10 is added to the extracted value.
chmode:	Channel mode 3 =^ interval counter (counter that is reset every time a measurement data record is created)
index:	Is used for the user-defined sorting of the fields in the selection lists

The third line describes the second measurement value "Height" in the measurement data container used:

Height:	"Height" should be used as the name for the measurement value.
f32:	The data type used for the measurement value should be a 32-bit float (i.e. 4 bytes).
title:	Name of the measurement value that is displayed on the server
decpl:	Number of decimal places that should be displayed
units:	Unit of the measurement value that is displayed on the server
min:	Minimum value for the further display on the server (e.g. graphic)

max:	Maximum value for the further display on the server (e.g. graphic)
vscale:	Virtual scaling of the value (see "Attributes of the field definition" on page 254). In the current example the extracted value is multiplied by 0.01.
chmode:	Channel mode 6 =^ analogue channel (a "simple" measurement value, e.g. the temperature)
index:	Is used for the user-defined sorting of the fields in the selection lists

Lines 4-6 describe the third measurement value "Pump" in the measurement data container used:

Pump:	"Pump" should be used as the name for the measurement value.
u8:	The data type used for the measurement value should be an 8-bit unsigned integer (i.e. 1 byte).
view:	Specifies from which user level the field is visible on the interface of the server 99 =^ field is not visible to anyone (required as in the current example the "Pump_ MSK" shadow field should be used instead of the "Pump" field. Shadow fields can be used to divide one byte field up into several bit fields).
edit:	Specifies the user level that is required to be able to change the field content via the interface of the server 99 =^ field cannot be changed by anyone (required as in the current example the "Pump_MSK" shadow field should be used instead of the "Pump" field).
@Pump:	Specifies that the "Pump_MSK" shadow field defined in line 6 should use the memory area of the "Pump" field defined in line 4
Pump_MSK:	"Pump_MSK" should be used as the name for the shadow field.
dlorw=skip:	The shadow field is not considered for the access functions to the container which are automatically generated for the device logic (in the current example histdata channel 0). I.e. within the device logic writing/reading of the shadow field must be done via manual masking of the desired bit in the "Pump" field.
bitmask:	Bit mask to mask the actual bits to be used out of the data block, hexadecimal In the current example, the least significant bit (LSB) is masked out of the "Pump" field.
min:	Minimum value for the further display on the server (e.g. graphic)
max:	Maximum value for the further display on the server (e.g. graphic)
chmode:	Channel mode. 1 =^ digital

The 7th line describes the fourth measurement value "Info" in the measurement data container used:

Info:	"Info" should be used as the name for the measurement value.
astr.50:	The data type used for the measurement value should be an ANSI string. After the dot the number of characters (50 in the current example) are specified.
title:	Name of the measurement value that is displayed on the server
index:	Is used for the user-defined sorting of the fields in the selection lists.

index is not specified for "Pump", thus "Pump" automatically receives index 2. The sorting order of the channels is therefore "Height", "Delay", "Pump", "Info".

16.3 Special values of the data types

Each numerical data type supports special states, such as NAN (Not a Number). If such a value is detected on the server, the standard display and further processing in myDatanet is applied.

Overview of the possible values (unsigned):

Value/type	u8 (byte)	u16 (word)	u32 (dword)
NaN	0xFF	0xFFFF	0xFFFFFFFF
OF	0xFE	0xFFFE	0xFFFFFFFE
UF	0xFD	0xFFFD	0xFFFFFFFD
OL	0xFC	0xFFFC	0xFFFFFFFC
SC	0xFB	0xFFFB	0xFFFFFFFB

Overview of the possible values (signed):

Value/type	s8 (bint)	s16(wint)	s32 (dint)	s64 (qint)
NaN	127	32767	2147483647	0x7FFFFFFFFFFFFFFF
OF	126	32766	2147483646	0x7FFFFFFFFFFFFFFE
UF	-126	-32766	-2147483646	0x8000000000000002
OL	-127	-32767	-2147483647	0x8000000000000001
SC	-128	-32768	-2147483648	0x8000000000000000

Overview of the possible values (float):

Value/type	f32 (float32)	f64 (float64)
NaN	0x7F800001	0x7FF0000000000001
OF	0x7F800002	0x7FF0000000000002
UF	0x7F800003	0x7FF0000000000003
OL	0x7F800004	0x7FF0000000000004
SC	0x7F800005	0x7FF0000000000005

Chapter 17 API

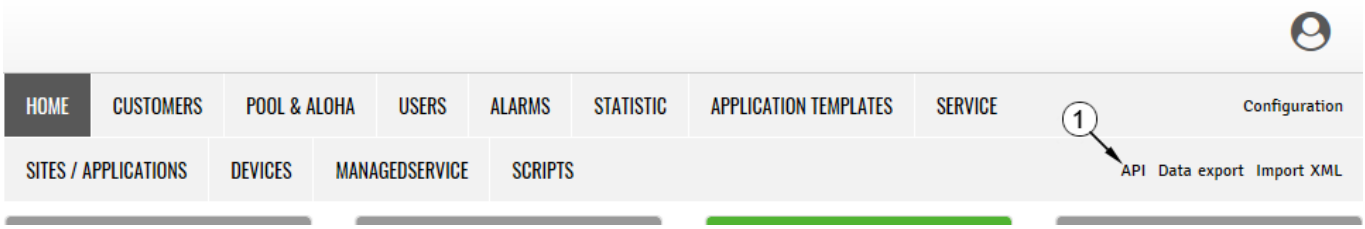
Important note: *The relevant licences are required on the myDatanet server to use the API (Application Programming Interface). For future information contact your responsible sales partner.*

17.1 Backend API

The API is provided to export data from and import data to the myDatanet server. However, this is not just limited to the pure measurement data but includes all of the data provided by myDatanet server (e.g. configurations). It is therefore possible for the customer to completely dispense with the interface of the myDatanet server and to create his own user interface. A specially developed PC program or web interface can, for example, be used for this purpose.

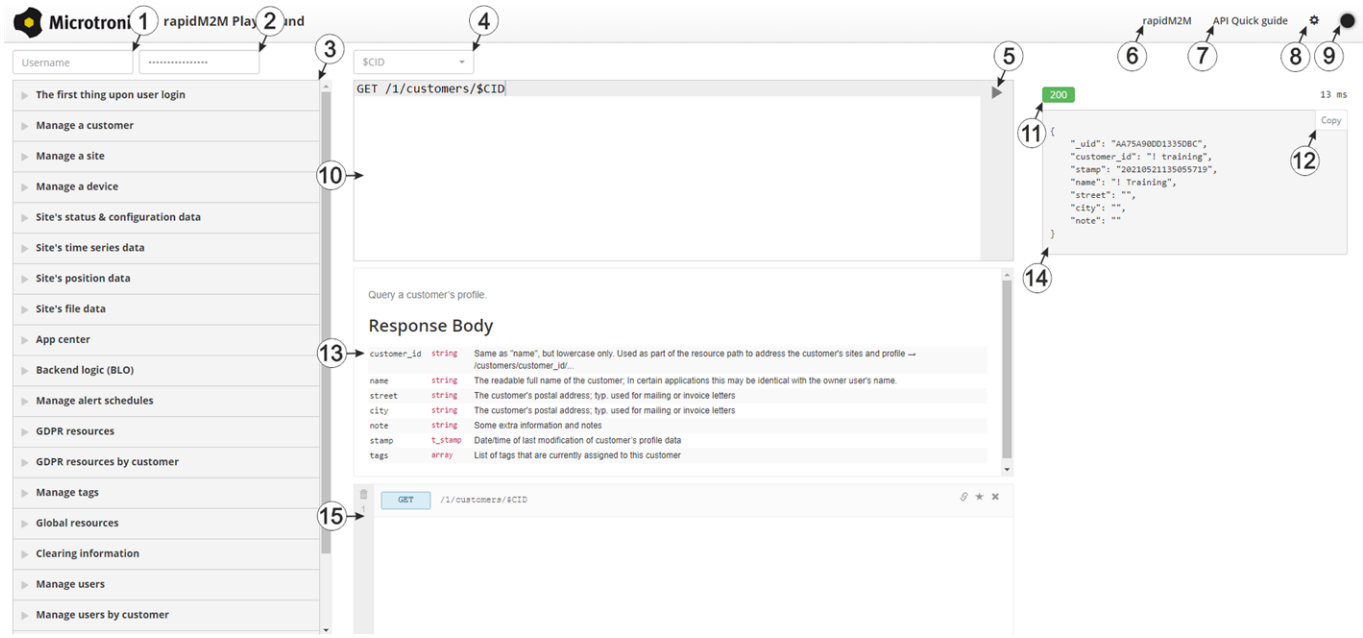
17.2 rapidM2M Playground

The rapidM2M Playground enables you to familiarise yourself with the API of the myDatanet server and to test the provided functions. One click on the "API" button will take you to rapidM2M Playground .



1	Opens the rapidM2M Playground

17.2.1 Overview



rapidM2M Playground

1	Input field for the user name
2	Input field for the password
3	List of the available HTTP commands. The HTTP commands are grouped according to their fields of application.
4	Depending on the selected HTTP command, the drop down lists for selecting the customer, user and site that should replace the corresponding wild cards ("$CID"...customer , "$UID"...user, "$SID"...site) in the resource path of the HTTP command are displayed.
5	Button to execute the HTTP command
6	Opens the website "http://rapidm2m.com/" that includes additional information for developers
7	Opens the quick guide for the API
8	Button for displaying the menu that contains the global settings
9	Button to change the colour scheme of the rapidM2M Playground
10	Window displaying the selected HTTP command
11	Response code sent by the myDatanet server as an answer to the HTTP command
12	Copies the JSON object generated as a response to the HTTP command on to the clipboard
13	Window displaying the documentation for the selected HTTP command. Depending on the selected command, this includes a description of the action being executed, information that must be observed and a description of the request body and response body.
14	Window displaying the JSON object that is generated as a response to the HTTP command
15	Window displaying the last executed HTTP commands

Chapter 18 Maintenance

Important note: *To prevent any damage to the device, the work described in this section of the instructions must only be performed by qualified personnel.*

The device must be deenergised before any maintenance, cleaning and/or repair work.

18.1 General maintenance

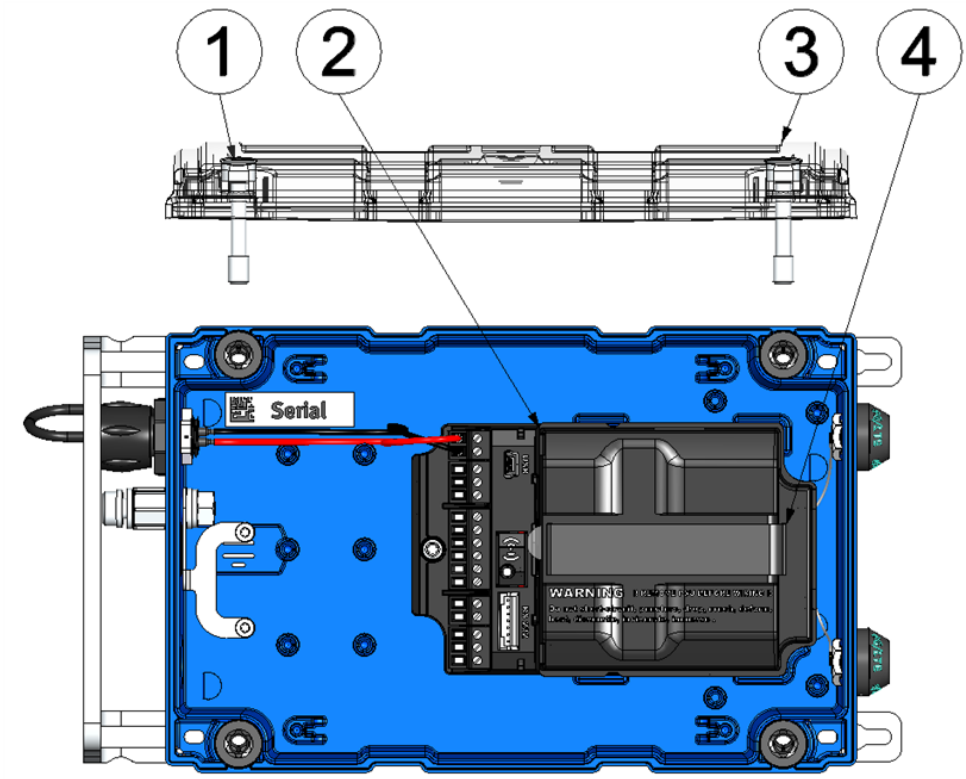
- Regularly check the myDatalogEASY IoT GW ATEX for mechanical damage.
- Check all of the connections for leaks or corrosion on a regular basis.
- Check all of the cables for mechanical damage at regular intervals.
- Clean the myDatalogEASY IoT GW ATEX with a soft, moist cloth. Use a mild cleaning agent, if necessary.

Important note: *Due to the electrostatic effects, the myDatalogEASY IoT GW ATEX must not be rubbed with cloths in the Ex zone.*

18.2 Replacing the power supply unit

Important note: *A dry location must be used to replace the power supply unit. If this is not possible, protect the opened device against penetrating moisture using suitable means.*

How-To-Video: Replacing the PSU and the silica gel pouch



Opening the myDatalogEASY IoT GW ATEX

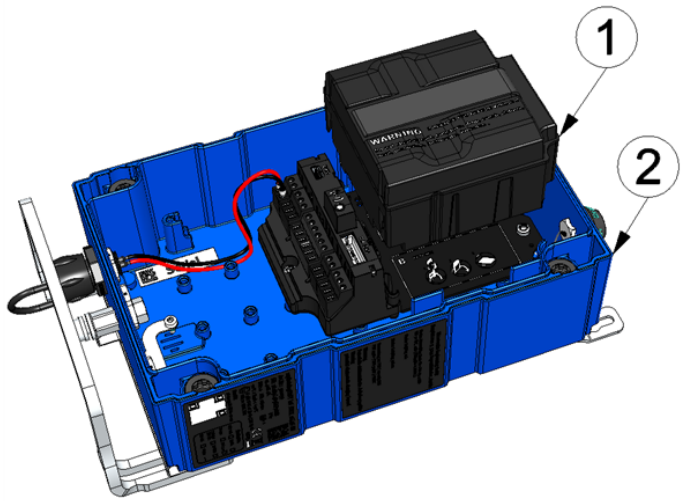
1 Hexagon socket screw M6x30	**3** Housing cover
2 Power supply unit	**4** Strap to remove the power supply unit

1. Ensure that all of the relevant data was transferred to the myDatanet server. If necessary, initiate a transmission. To do so, execute the operations provided in the Device Logic and then check again that all of the relevant data has been transferred.
2. If you are using an external supply or charging voltage, disconnect this from the device before opening the housing cover.
3. Remove the four screws that secure the housing cover. Now open the myDatalogEASY IoT GW ATEX .

4. Remove the power supply unit from the myDatalogEASY IoT GW ATEX and replace the existing power supply unit with a new one. Use the strap provided to remove the power supply unit.

   Important note: *If the pressure compensation has been sealed via the Gas protection set for myDatalogEASY IoT series (301414) (see "Sealing the pressure compensation" on page 58), the silica gel pouch also has to be replaced when replacing the PSU.*

   Note: *Ensure that power supply units, especially ones with integrated energy store (rechargeable battery or battery), are disposed of in line with environmental requirements. Power supply units with depleted rechargeable battery or battery can be returned to the manufacturer or handed in at suitable collection points.*



Removing the power supply unit

1 Power supply unit	**2** myDatalogEASY IoT GW ATEX base unit

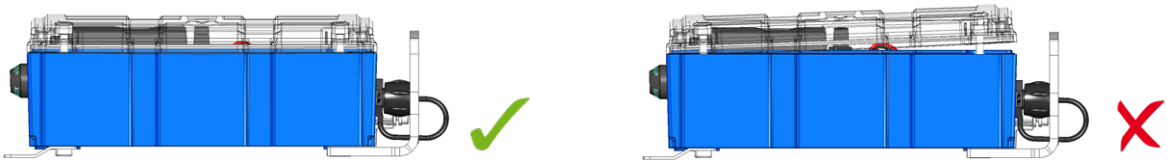
The following step is not mandatory.

5. Check whether the connection to the myDatanet functions correctly (see "Testing communication with the device" on page 81).
6. Close the housing cover. The best option is to tighten the four screws crosswise (torque: max. 1Nm) so that the housing cover is positioned evenly.

   Important note: *Ensure that the seals are clean and intact before closing the housing cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.*

7. Check that the housing cover is positioned correctly on all sides and that no foreign materials have been trapped between the housing and housing cover.

   Important note: *The manufacturer is not liable for any damage that is caused by housing covers that are not closed correctly.*



The following step is only necessary if you are using an external supply or charging voltage.

8. Now switch on the external supply or charging voltage.

   Note: *If you are using a power supply unit without an integrated energy store, the external supply or charging voltage must be switched on before the optional step during which the connection to the server is tested.*

Chapter 19 Removal/disposal

Incorrect disposal can cause environmental hazards.

Dispose of the device components and packaging material in accordance with the locally valid environmental regulations for electronic products.

1. Disconnect any charging voltage that has been used.
2. Remove the power supply unit with the integrated energy storage (non-rechargeable battery) and dispose of it separately.
3. Disconnect any connected cables using a suitable tool.



Logo of the EU WEEE Directive

This symbol indicates that the requirements of Directive 2012/19/EU regarding the scrap disposal of waste from electric and electronic equipment must be observed. Microtronics Engineering GmbHsupports and promotes recycling and environmentally friendly, separate collection/disposal of waste from electric and electronic equipment in order to protect the environment and human health. Observe the local laws and regulations on disposal of electronic waste at all times.

Microtronics Engineering GmbHreleases goods brought onto the market in Austria from the obligations via ERA, which means that collection points that cooperate with ERA Elektro Recycling Austria GmbH (https://www.era-gmbh.at/) can be used for disposal in Austria.

The device contains a lithium button cell that has been soldered on. It must be removed before disposal or the disposal service must be informed that batteries are still located in the device.

The device includes a battery or rechargeable battery (lithium) that must be disposed of separately.

Chapter 20 Troubleshooting and repair

20.1 General problems

Problem	Cause/solution
Device does not respond.	• Check the cable connections (see "Connection of the power supply" on page 71) • The capacity of the energy store in the power supply unit is depleted.
Communication problems	• Load the device log from the myDatalogEASY IoT GW ATEX using the DeviceConfig (see ""Log" tab" on page 101). A list of all the possible error codes is included in the chapter "Log entries and error codes" (see "Log entries and error codes" on page 272). • The capacity of the energy store in the power supply unit is virtually depleted.
The connection via the external SIM card is not working.	• The chargeable feature "Activation code VPN SIM (300539)" has not been released. • Check whether the external SIM card has been inserted correctly (see "Inserting/replacing the SIM card" on page 56). • Check whether the configuration data (PIN, APN, username and password) have been set correctly via the "rM2M_SetExtSimCfg()" function. • Check whether the configuration data (PIN, APN, username and password) have been set correctly via the DeviceConfig . • Check whether the PIN code (if required by the SIM card) has been set correctly by the DeviceConfig .
Not all or no data is available on the server.	• The connection was aborted during the transmission, which is indicated by a time-out entry in the connection list (see "myDatanet Server Manual " 805002). Solution: Initiate a transmission or wait for the next cyclical transfer. • The Data Descriptor was not configured correctly (see "Data Descriptor " on page 251). • The assignment of the device and site is not correct (see "Site" on page 84).
The Device Logic is not being executed correctly.	• Check that the correct Device Logic type was selected during the configuration of the control (see "Control" on page 85). • Load the device log from the myDatalogEASY IoT GW ATEX using the DeviceConfig (see ""Log" tab" on page 101). A list of all the possible Device Logic error codes is included in the chapter "Device logic error codes" (see "Device Logic error codes" on page 226). • The previous Device Logic has been replaced with that of the newly assigned site/application due to a context change (assignment of a different site/application). • By assigning a new or different site/application, the Device Logic installed via the USB has been replaced with that of the newly assigned site/application.

20.2 Log entries and error codes

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1000	POWER ON	0	---	Restart following a power failure
		4	---	Watchdog reset (e.g. because of an exception)
		6	---	Reset was initiated by the device itself (e.g. in event of firmware update)
		##	--	Restart for another reason. There may be a hardware problem if the "POWER ON" log entry with a parameter code that is not equal to 0 or 6 is contained in the device log several times. Contact the manufacturer in this case (see "Contact information" on page 293).
1030	UV LOCKOUT	---	---	The device switches to energy saving mode and terminates all of the operations as the rechargeable battery or battery voltage is too low. Only the charge controller, if present, remains active.
1031	UV RECOVER	---	---	The rechargeable battery or battery voltage once again suffices to guarantee reliable operation. This is achieved by replacing the rechargeable battery or battery pack.The device resumes normal operation in accordance with the configuration.
1034	CONTROLLER UPDATE	##	---	Controller firmware update was completed successfully This entry is always duplicated in the device log. In the first entry, the parameter specifies the major version number (e.g. 3 for 03v011), while in the second entry it specifies the minor version number (e.g. 11 for 03v011).
1035	EXCEPTION	##	---	An internal system error was detected that caused the device to restart. The parameter specifies the type of system error. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 293).
1038	UV MODEM LOCKOUT	---	---	The device deactivates the modem because the rechargeable battery or battery voltage is too low. A connection cannot be established now.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1039	UV MODEM RECOVER	---	---	The rechargeable battery or battery voltage once again suffices to guarantee a stable connection. This is achieved by replacing the rechargeable battery or battery pack.
1161	LOG REFORMATFILE	##	---	Errors in file system have been resolved. This can result in data being lost (data and/or log entries). The parameter contains more information on the problem. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 293).
1192	FUTURE TIMESTAMP	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 293).
1200	MODEM ERROR			Modem error (see "Modem error" on page 277)
1201	MODEM NOT FOUND	---		Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 293).
1202	MODEM CMME ERROR	##	---	The GPRS modem indicates a +CME error. The parameter specifies the type of error.
1203	SELECTED NETWORK	##	---	A new GSM network was selected. The parameter specifies the MCC (Mobile Country Code) and the MNC (Mobile Network Code) of the selected GSM network.
1207	GSM NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1208	GPRS NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming
1212	ERROR MODEM IRREGULAR OFF	##	---	Indicates a faulty connection. The parameter includes a counter that indicates how many consecutive connections have not worked.
1219	LTE NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming
1252	MODEM TO CON	##	---	Timeout while a connection is being established. The parameter specifies the reason for the timeout. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 293).
1281	ZLIB STREAMPROCESS ERR	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 293).
1282	ZLIB STREAMFINISH ERR	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 293).
1300	USB CONNECTED	---	---	USB connection to a PC established.
1310	USB DISCONNECTED	---	---	USB connection was disconnected.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1317	BLE CONNECTED	---	---	Bluetooth connection to a PC established
1318	BLE DISCONNECTED	---	---	Bluetooth connection was terminated
1335	LOG_SHT2X_ STATE	0	SHT2X SENSOR OK	The internal temperature and air humidity sensor is returning valid values again
		1	SHT2X RH ERROR	A communication error occured when reading the air humidity value from the internal temperature and air humidity sensor.
		2	SHT2X TEMP ERROR	A communication error occured when reading the temperature value from the internal temperature and air humidity sensor.
		3	SHT2X RH+TEMP ERROR	A communication error occured when reading the measurement value from the internal temperature and air humidity sensor.
		4	SHT2X PLAUSIBILITY ERROR	The values received from the internal temperature and air humidity sensor are not plausible (rH <0% rH or >100% rH or temperature <-40°C or >125°C)
1336	SHT2X COM ERR	---	---	Communication with the internal temperature and air humidity sensor is not possible (sensor not present or faulty)
1337	SHT2X COM ERR1	---	---	Starting the internal temperature measurement failed
1338	SHT2X COM ERR2	---	---	Starting the internal air humidity measurement failed
1339	SHT2X TEMP RAW	##	---	Temperature raw value (register value from the internal temperature and air humidity sensor) if a plausibility error (SHT2X PLAUSIBILITY ERROR) was detected
1340	SHT2X RH RAW	##	---	Air humidity raw value (register value from the internal temperature and air humidity sensor) if a plausibility error (SHT2X PLAUSIBILITY ERROR) was detected
1601	SIM_STATE	0	NONE	SIM state was changed to "NONE" (initial state).
		1	PRODUCTION	SIM state was changed to "PRODUCTION" (a new device is in stock).
		2	HOT	SIM state was changed to "HOT" (valid contract).
		3	COLD	SIM state was changed to "COLD" (end of contract or fair use policy violated).
		4	DISCARDED	SIM state was changed to "DISCARDED" (device has been decommissioned).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1602	EXTERNAL SIM	-2	NOT ALLOWED	Establishing the connection via the external SIM card is not permissible • No APN settings (APN, username and password) saved in the device • Use of the external SIM slot is not released
		-1	NOT FOUND	The external SIM card is not present or could not be accessed.
		0	OK	The external SIM card could be accessed when establishing the connection. However, this log entry does not indicate whether the connection itself was established successfully.
1910	ACCU 0 E2PROM ERROR	0	---	Rechargeable battery not available
		1	---	Invalid length of the data structure in the EEPROM of the rechargeable battery
		2	---	No charging profile available in the EEPROM (only with Li-ion rechargeable batteries)
		3	---	Error when reading the SoC-value
		4	---	Error when writing the SoC-value
		5	---	The charging profiles of the rechargeable batteries inserted do not match (only with devices that support the simultaneous use of multiple rechargeable batteries)
		6	---	• Permissible charging time exceeded • When restarting the device, it was recognised that the rechargeable battery currently in use has already exceeded the permissible charging time once. The battery is probably defective and should be checked by the manufacturer.
2000 - 2199	MODULE ERR	##	---	Area for customer-specific critical error codes, that can be written in the device log by means of the "rM2M_WriteLog()" function
2200 - 2399	MODULE WARNING	##	---	Area for customer-specific non-critical error codes, that can be written in the device log by means of the "rM2M_WriteLog()" function
2400 - 2599	MODULE INFO	##	---	Area for customer-specific information about the current operating state, that can be written in the device log by means of the "rM2M_WriteLog()" function

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
2600 - 2799	MODULE DEBUG	##	---	Area for customer-specific debug information, that can be written in the device log by means of the "rM2M_WriteLog()" function
3000 - 3099	SCRIPT ERROR	##	--	Error codes of the script execution (see "Device Logic error codes" on page 226).

20.2.1 Modem error

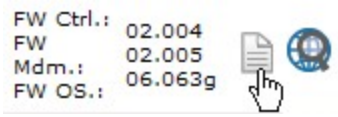
Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
External SIM card				
1200	SIM PIN NO ATTEMPT	-999	---	The PIN code transferred to the system via the "rM2M_SetExtSimCfg()" function is not correct. Another input attempt is not made to ensure the SIM card is not locked.
1200	SIM ERROR	-998	---	Error when accessing the external SIM card • SIM card not recognised. • Use of the external SIM slot is not released
GPRS error				
1200	BEARER GPRS FAILED	-988	---	GPRS setup error • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.com/footprint).
1200	BAND SEL FAILED	-969	---	A network could not be found on the GSM900/1800 or on the GSM850/1900 band. • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.com/footprint).
External SIM card				
1200	SIM PIN WRONG	-968	---	The PIN code transferred to the system via the "rM2M_SetExtSimCfg()" function is not correct.
	SIM NO PIN	-967	---	No PIN code was transferred to the system via the "rM2M_SetExtSimCfg()" function. However, the SIM card requires a PIN code to be entered.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1200	NETLOCK ERROR	-966		Error when selecting the network. Check whether the device is in the coverage area. Internal SIM chip: see www.microtronics.com/footprint External SIM card: Contact the provider that supplied the SIM card.
TCP channel error				
1200	CHANNEL ABORTED	-965	---	An attempt is being made to write to/read a TCP client that is no longer available. Try again later
	TCP DNS FAILURE	-958	---	The name could not be resolved in an IP address. Internal error
	CHANNEL REFUSED	-955	---	The TCP connection has been refused by the server. Try again later
	CHANNEL HOST UNREACHABLE	-954	---	No route to the host. Try again later
	CHANNEL NETWORK UNREACHABLE	-953	---	No network available Try again later
	CHANNEL PIPE BROKEN	-952	---	TCP connection interrupted Try again later
	CHANNEL TIMEOUT	-951	---	Timeout (DNS request, TCP connection, ping response, etc.) Try again later
	MODEM POSITION UPDATE ERROR	-943	---	Timeout during determination of the GSM position data

20.3 Evaluating the device log

20.3.1 Evaluating the device log on the myDatanet server

The last 300 log entries on the myDatanet server can be called up via the button shown below that is located in the measurement device list. As the log entries are sent to the server in the transmission cycle in the same way as the measurement data, only the log entries up to the last server connection are available.



The manual for the server ("myDatanet Server Manual " 805002) includes a detailed description of the evaluation of the device log on the myDatanet server.

20.3.2 Evaluating the device log using DeviceConfig

The DeviceConfig program can be used to read all of the stored log entries, including those that have not yet been transferred to the myDatanet server, directly from the myDatalogEASY IoT GW ATEX via the USB interface or the Bluetooth interface.

A more detailed description about the evaluation of the device log using DeviceConfig is included in chapter ""Log" tab" on page 101.

Chapter 21 Spare parts and accessories

21.1 Chargeable features

Description	Quantity	Order number
Order option (feature is activated by the manufacturer prior to delivery)		
Feature activation VPN SIM	1	300729
Activation code (for later activation by the customer)		
Activation code VPN SIM	1	300539

21.2 Assembly sets

Description	Quantity	Order number
Housing ATEX 300x200x150mm for outdoor mounting	1	301238
Pipe mounting set for housing 300x200x150mm	1	301184
Wall mounting set for housing 300x200x150mm	1	301185
Pipe mounting EASY IoT 30-250 mm	1	301191
Niro shackle	1	206.325
Anchor clamp 5,5 - 10,5mm	1	301017

21.3 Antennas

Description	Quantity	Order number
Portable antenna multi band FME-F	1	206.826
Flat antenna Disc multi band 2xFME-F 2m	1	301196
Dome antenna multi band FME-F 3m	1	301211
Extension cable for antenna FME-F/FME-M 5m	1	206.805

21.4 Power supply units

Description	Quantity	Order number
Battery packs		
PSU713 BP (Li-SOCl2; 13Ah; -20...+50°C operating temperature)	1	300526
Direct supply		
PSU DC (-20...+60°C operating temperature)	1	300529

21.5 Charging devices and power supply units

Description	Quantity	Order number
Snap-on primary plug EU for chargers/power supplies	1	300027
Snap-on primary plug UK for chargers/power suppliest	1	300028
Snap-on primary plug US for chargers/power supplies	1	300029
Snap-on primary plug AUS for chargers/power supplies	1	300030
PSU Charger	1	300697

21.6 Solar panel

Description	Quantity	Order number
PV module 30W for ATEX myDatalogEASY IoT series	1	301208

21.7 Cable

Description	Quantity	Order number
Connection cable 2-pins for supply 5m	1	301192
Clamping tube	1	300256

21.8 BLE sensors

Description	Measurement range	Quantity	Order number
BLE Gauge pressure sensor 0-3m 9W	0-3m	1	300893+300871
BLE Gauge pressure sensor 0-1m 9W	0-1m	1	300893+300872
BLE Gauge pressure sensor 0-10m 9W	0-10m	1	300893+300891
BLE Radar	0...5m	1	301443

21.9 BLE output modules

Description	Quantity	Order number
BLE mA Link	1	300870

21.10 Other accessories

Description	Quantity	Order number
MDN Magnet	1	206.803
USB BLE-Adapter	1	300685
Gas protection set for myDatalogEASY IoT series	1	301414

Chapter 22 Document history

Rev.	Date	Changes
01	24.06.2021	**First version**
02 (1/3)	29.06.2022 (1/3)	**Hardware version 1.1** *The device is now equipped with a metal bracket to protect the connectors* **Chapter "Declaration of conformity" on page 13** *Declaration of Conformity updated* **Chapter "Specifications" on page 19** *Specification of dimensions and weight adapted for hardware version 1.1.* *Ex certification changed from "<Ex> II 3G Ex eC IIB T4 Gc" to "<Ex> II 3G Ex ec nC IIC T4 Gc"* *Storage temperature changed from -30...+85°C to -20...+50°C (corresponds to the operating temperature)* *Number of maximum available antenna connectors reduced from 3 to 2* *Specification of the supported frequency bands for the myDatalogEASY myDatalogEASY IoT GW ATEX 2G/M1/NB1 Worldcorrected.* **Chapter "Ex protection" on page 24** *Note added that the connectors must not be separated when energized or when there may be an explosive atmosphere.* *Note added that the transient protection (according to EN 60079-7) is ensured by the Connection cable 2-pins for supply 5m (301192).* *Ex certification changed from "<Ex> II 3G Ex eC IIB T4 Gc" to "<Ex> II 3G Ex ec nC IIC T4 Gc"* **Chapter "Overview" on page 28** *Requirements concerning the earth cable added* **Chapter "Intended use" on page 30** *Data transmission via LPWAN is no longer supported.* **Chapter "General product information" on page 30** *Data transmission via LPWAN is no longer supported.* **Chapter "Device labelling" on page 32** *The order option "Feature activation LPWAN " is no longer available* *Ex certification changed from "<Ex> II 3G Ex eC IIB T4 Gc" to "<Ex> II 3G Ex ec nC IIC T4 Gc"* **Chapter "Functional principle" on page 37** *The direct transmission of data to a customer-specific server via "LPWAN" or "NB-IoT" is no longer available.* **Chapter "Determining the transmission paths"** *Chapter removed* **Chapter "Procedure in case of connection aborts" on page 42** *Explanation of the process when a connection is aborted during the device logic download, revised* **Chapter "Timeout monitoring in online mode" on page 43** *Chapter added*

Rev.	Date	Changes
02 (2/3)	29.06.2022 (2/3)	**Chapter "File transfer" on page 46** *The number of files that can be registered for the file transfer has been increased from 20 to 60.* **Chapter "Storage" on page 52** *Storage temperature changed from -30...+85°C to -20...+50°C (corresponds to the operating temperature)* **Chapter "Scope of supply" on page 51** *The brackets for wall mounting are no longer included in the scope of delivery as they are now pre-mounted to the base unit.* **Chapter "Dimensions" on page 55** *Specification of dimensions adapted for hardware version 1.1.* **Chapter "Wall mounting" on page 62** *Description of the installation adapted for hardware version 1.1.* **Chapter "Connection of the power supply" on page 71** *Requirements concerning the earth cable added* *For connecting the supply or charging voltage now the Connection cable 2-pins for supply 5m (301192) is recommended.* *Note added that the myDatalogEASY IoT GW ATEX must be connected to the ground potential of the system.* *Note added that the connection point between the connection cable and the power source must be outside the hazardous area.* *Note added that unused plugs must be sealed waterproof by means of protective caps.* **Chapter "Connecting the mobile network antennas" on page 74** *Number of maximum available antenna connectors reduced from 3 to 2* **Chapter "Measurement instrument" on page 88** *Explanation of the "Modem Version" and "OS Version" fields that are no longer used removed* **Chapter "Functional principle" on page 92** *Note added that the USB interface is a service interface.* **Chapter "Connecting a Device via Bluetooth Low Energy" on page 98** *Chapter added* **Chapter "Recommended procedure" on page 107** *Chapter added* **Chapter "Uplink" on page 138** *Explanation of the arrays with symbolic indices "TrM2M_GSMInfo" extended to include the description of the "act", "lac" and "cid" elements* *Explanation of the array with symbolic indices "TrM2M_TxItfStats" added* *Explanation of the constants for the mobile radio AcT (access technology) added* *Explanation of the constants for the signal strength measurement flags added* *Explanation of the "rM2M_TxItfGetStats()" and "rM2M_SetTCPKeepAlive()" functions* *Explanation of the "rM2M_GSMGetRSSI()" and "rM2M_GetRSSI()) functions extended to include the description of the "flags" parameter*

Rev.	Date	Changes
02 (3/3)	29.06.2022 (3/3)	**Chapter "Bluetooth Low Energy" on page 161** *Chapter added* **Chapter "Char & String" on page 192** *Explanation of the "memcpy_native", "memset_native" and "memcomp_native" functions added* **Chapter "Various" on page 201** *Explanation of the "delay_us()" function added* **Chapter "File transfer" on page 212** *Explanation of the "FT_CMD_ENUM" and "FT_CMD_RETR" file transfer commands added* *Explanation of the callback function, that must be provided by the script developer, extended to include a description of how the file transfer commands "FT_CMD_ENUM" and "FT_CMD_RETR" should be handled.* *Explanation of the "FT_RegisterEnum()" function added* **Chapter "Device Logic error codes" on page 226** *Explanation of error code "SCRIPT_ERR, SCRIPT UPDATE ERROR" revised* *Explanation of error codes"SCRIPT_ERR, SCRIPT SYSTEM SHUTDOWN", "SCRIPT_ERR, SCRIPT DOWNLOAD ERROR" und "SCRIPT_ERR, SCRIPT DELETED" added* *Explanation of error codes "LOG_NOSCRIPT_ERR, SCRIPT xxx" added* **Chapter "Log entries and error codes" on page 272** *Explanation of the error codes "GSM NETWORK REGISTRATION", "GPRS NETWORK REGISTRATION","LTE NETWORK REGISTRATION", "SHT2X SENSOR OK", "SHT2X RH ERROR", "SHT2X TEMP ERROR", "SHT2X RH+TEMP ERROR", "SHT2X PLAUSIBILITY ERROR", "SHT2X COM ERR", "SHT2X COM ERR1", "SHT2X COM ERR2", "SHT2X TEMP RAW" und "SHT2X RH RAW" added* **Chapter "Chargeable features" on page 281** *The features "LPWAN" and "NB-IoT" for the direct transmission of data to a customer-specific server are no longer available.* *The feature "Cellular TCP IP" is no longer available because this function is now enabled by default.* **Chapter "Antennas" on page 281** *LPWAN antennas removed* *Accessories list revised* **Chapter "Solar panel" on page 282** *Chapter added* **Chapter "Cable" on page 282** *Accessories list revised*
03 (1/2)	01.08.2022 (1/2)	**Chapter "Specifications" on page 19** *Explanation regarding the PSU413D+ AP removed* **Chapter "Ex protection" on page 24** *PSU413D+ AP removed from the list of admissible power supply units* **Chapter "Block diagram" on page 29** *Note removed that a PSU containing a rechargeable battery is recharged via the V Batt output of the DC/DC converter*

Rev.	Date	Changes
03 (2/2)	01.08.2022 (2/2)	**Chapter "Storage" on page 52** *Information and note regarding the PSU413D+ AP removed* **Chapter "Transporting power supply units" on page 52** *Information regarding the PSU413D+ AP removed* **Chapter "Technical details about energy management" on page 75** *Explanations regarding the PSUs with rechargeable batteries removed* **Chapter "Technical details about the energy supply" on page 76** *Explanation regarding the PSU413D+ AP removed* **Chapter "Commissioning the system" on page 79** *Note regarding PSUs with rechargeable energy stores removed* **Chapter "Testing communication with the device" on page 81** *Note regarding PSUs with rechargeable energy stores removed* **Chapter "Charging the power supply unit"** *Chapter removed* **Chapter "Power supply units with integrated energy store"** *Chapter removed* **Chapter "Power supply units" on page 281** *Information regarding the PSU413D+ AP removed*
04 (1/2)	26.01.2023 (1/2)	**Chapter "Declaration of conformity" on page 13** *Declarations of Conformity updated* **Chapter "Ex certification" on page 17** *Chapter with the EC-type examination certificate added* **Chapter "Specifications" on page 19** *Protective circuit integrated in the PSU DC specified in more detail* *Protection class adjusted from IP66 to IP66 / IP68* *B71 removed from the list of supported LTE frequency bands for the variante myDatalogEASY IoT GW ATEX 2G/M1/NB1 World* **Chapter "Device labelling" on page 32** *Type plate updated* **Chapter "Scope of supply" on page 51** *Scope of delivery extended to include the "Gas protection set for myDatalogEASY IoT series (301414)"* **Chapter "Technical details about the energy supply" on page 76** *Protective circuit integrated in the PSU DC specified in more detail* **Chapter ""Customer" area" on page 118** *Screenshots of the user interface of the myDatanet server adapted to version 50v007* **Chapter ""Sites / Applications" area at customer level" on page 120** *SScreenshots of the user interface of the myDatanet server adapted to version 50v007* **Chapter "Constants" on page 132** *Returncode "ERROR_SENSOR_DISABLED" added*

Rev.	Date	Changes
04 (2/2)	26.01.2023 (2/2)	**Chapter "Timer, date & time" on page 133** *Explanation of the arrays with symbolic indices "TrM2M_DateTime" extended* *Explanations of the "rM2M_TimerAdd()" and "rM2M_TimerAddExt()" functions extended* **Chapter "Uplink" on page 138** *Explanation of the function, that must be provided by the device logic developer and that is called up from the internal flash memory after reading a data record, extended to include the description of the parameter "timestamp256"* *Explanation of the "rM2M_CfgRead()" function extended* **Chapter "Encoding" on page 155** *Explanation of the "rM2M_SetPacked()" function corrected, now indicating that in connection with signed data types problems can arise and not in connection with the formerly indicated unsigned data types.* *Explanation of the function "rM2M_Pack()" extended* **Chapter "Bluetooth Low Energy" on page 161** *Explanation of the BLE error codes revised* *Note added to the "BLE_Setbuf ()" function, indicating that the buffers "rxbuf" and "txbuf" have to be valid throughout the use by the firmware* **Chapter "Registry" on page 175** *Explanation of the constants for the "Indices of the registration memory blocks" extended* *Explanations of the "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()" and "rM2M_RegOnChg()" functions extended* **Chapter "Char & String" on page 192** *Explanations of the "sprintf()", "strcat()", "strcmp()", "strcspn()", "strpbrk()", "strstr()", "strtol()" and "atof()" funcitons extended* **Chapter "CRC & hash" on page 200** *Explanation of the "MD5()" function extended* **Chapter "Various" on page 201** *Explanations of the "getapilevel()", "exists()", "rtm_start()", "funcidx()", "numargs()" and "getarg()" functions extended* **Chapter "Console" on page 208** *Explanation of the "printf()" function extended* *Note added to the "setbuf()" function, indicating that the buffer "buf" has to be valid throughout the use by the firmware* **Chapter "Log entries and error codes" on page 272** *Explanation of the error code "ACCU 0 E2PROM ERROR" added* **Chapter "Other accessories" on page 282** *List of accessories extended to include the "Gas protection set for myDatalogEASY IoT series (301414)"*
05 (1/3)	28.07.2023 (1/3)	**Chapter "Declaration of conformity" on page 13** *Declaration of conformity updated* *Variant myDatalogEASY IoT 3G World removed* *Variant myDatalogEASY IoT 2G/3G/4G World added*

Rev.	Date	Changes
05 (2/3)	28.07.2023 (2/3)	**Chapter "Specifications" on page 19** *Specification added indicating which conditions have to be met (connections plugged / unplugged) for the respective protection class to apply.* *Variant myDatalogEASY IoT 3G World removed* *Variant myDatalogEASY IoT 2G/3G/4G World added* **Chapter "Ex protection" on page 24** *Note added indicating that when installing the device, the requirements of the standard for planning, the selection and construction of electrical installations in potentially explosive atmospheres as per (IEC 60079-14) have to be met.* **Chapter "Safety and precautionary measures for the GSM/GPRS modem installation " on page 27** *Note removed indicating that the device may not be operated in the vicinity of highly flammable areas or of flammable gases, vapours or dusts.* **Chapter "Scope of supply" on page 51** *Link and QR code referring to the How-To-Video "Unpacking the myDatalogEASY IoT GW ATEX " added.* **Chapter "Inserting/replacing the SIM card" on page 56** *Link and QR code referring to the How-To-Video "Inserting the SIM card" added.* **Chapter "Sealing the pressure compensation" on page 58** *Chapter added* **Chapter "Installing the myDatalogEASY IoT GW ATEX " on page 61** *Note added indicating that metallic mounting fixtures have to be grounded.* **Chapter "Pipe mounting" on page 63** *Chapter added* **Chapter "Outdoor installation" on page 64** *Chapter added* **Chapter "Supply concepts" on page 67** *Chapter added* **Chapter "Connection of the power supply" on page 71** *Symbols added to explain the correct connection of the supply in more detail.* **Chapter "Uplink" on page 138** *Explanation of the return value of the "rM2M_CfgWrite()" function rectified* **Chapter "Bluetooth Low Energy" on page 161** *Explanation of the return values of the "BLE_Init()" function extended to include the return code "ERROR_FEATURE_LOCKED"* *Explanation of the "BLE_Close()" function adjusted to account for the fact that when calling it up, the connection to the currently connected sensors is also disconnected.* *Explanation of the return values of the "BLE_Scan()", "BLE_Connect()", "BLE_Disconnect()", "BLE_GetConnState()", "BLE_Write()", "BLE_Read()" and "BLE_ChgConItv()" functions extended to include the return codes "ERROR-1" and "ERROR-2".* *Explanatiion of the "BLE_SetScanResponseData()" function added*

Rev.	Date	Changes
05 (3/3)	28.07.2023 (3/3)	**Chapter "Char & String" on page 192** *Note indicating that strings > 128 bytes are not supported removed from the explanations of the "strchr()", "strrchr()", "strspn()", "strcspn()","strpbrk()" and "strtsr()" functions.* **Chapter "Power management" on page 223** *Explanation of the flag "PM_FLAG_AKKU_FAULT" added, which is used for to signal the "Status of the power management".* *Explanation of the return value of the "PM_SetChargingMode()" function extended.* **Chapter "Differences to C" on page 248** *Note added indicating that variables are automatically initialised with "0".* **Chapter "Data structure" on page 251** *Explanation of the attribute "editmask" revised* **Chapter "Replacing the power supply unit" on page 266** *Links and QR codes referring to How-To-Videos added.* *Note added indicating that, if the pressure compensation had been sealed, the silica gel pouch also always has to be replaced when replacing the PSU.* **Chapter "Log entries and error codes" on page 272** *Explanation of the error code "MODEM NOT FOUND" added* **Chapter "BLE sensors" on page 282** *BLE Radar added*

Chapter 23 Glossary

App centre

Area of the myDatanet server for the installation and management of the IoT apps. The app models that serve as a basis for the IoT apps are obtained via the rapidM2M Store . When installing an IoT app on the myDatanet server the default settings defined when developing the app models are initially applied. These default settings can then be adjusted. Any number of IoT apps can be created based on a single app model by setting the appropriate default settings.

App model

An app model is developed in the rapidM2M Studio and forms the basis for creating IoT apps. It essentially contains the executable program files (device logic, backend logic, portal view, etc.) from which an IoT is created by adding the default settings. Distribution to the individual myDatanet servers is carried out via the rapidM2M Store . The available app models are displayed in the app centre of the respective myDatanet server.

Footprint

The manufacturer's devices are equipped with subscriber identity modules (SIM) ex-works for the purpose of mobile data transmission. The footprint describes those countries and regions where a mobile connection is available (see www.microtronics.com/footprint).

Device logic

The device logic is the intelligence installed on the device that determines the local functionality of the device. The device logic is part of the app model and is created in a C-like scripting language built on "PAWN".

Hardware ID string

Specifies the hardware platform installed in the device and its hardware version (e.g. rapidM2M M2 HW1.4). The part of the hardware ID string, that specifies the hardware version, is only increased if changes relevant to the rapidM2M firmware have been made to the hardware platform. When developing an app model, it can be specified on which hardware platform the app model can be installed and which version of the hardware platform is required as a minimum. The hardware ID string is displayed in the TESTbed of the rapidM2M Studio or in the "Identification" field of the input screen for configuring the device.

IoT app

IoT apps form the basis for creating sites. They consist of an app model and corresponding default settings that are applied as default values for the site when the site is created. The app centre can be used to create any number of IoT apps based on a single app model by setting the appropriate default settings. This makes sense if several use cases need to be covered by a single app model and they each require a different default site configuration (e.g. if a data logger with different external sensors is to be sold as a package).

NaN value

The myDatanet uses special encoding to display different error statuses in the measurement values, for example. By setting a measurement value to "NaN", it is clearly marked as invalid and is thus not used for any further calculations. In the measurement value graphs, a measurement value that has been set to "NaN" is indicated by an interruption in the graph. When downloading the data, a measurement value set to "NaN" is indicated by an empty data field.

Product revision

Specifies the revision of the product. The revision is increased every time the product is modified (i.e. electronic system, mechanics, etc.) and is marked on the type plate of the product.

rapidM2M Store

Is responsible for distributing the app models to the individual myDatanet servers. When installing and updating IoT apps the myDatanet server access the app models provided in the rapidM2M Store . The developer of the respective app model defines which myDatanet servers are allowed to access an app model via the rapidM2M Studio .

rapidM2M timestamp

Depending on the required accuracy, one of two special encodings can be used for the time stamp in rapidM2M. If the accuracy requirements are moderate, the "stamp32" data type (seconds since 1999-12-31 00:00:00 UTC) can be used. If a higher accuracy is required, the "stamp40" data type (1/256 seconds since 1999-12-31 00:00:00 UTC) can be used. Converting the "stamp32" data type into the UNIX timestamp (seconds since 1970-01-01 00:00:00 UTC) can be achieved by adding 946598400.

Chapter 24 Contact information

Support & Service:
Microtronics Engineering GmbH
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 7718023
support@microtronics.com
www.microtronics.com

Microtronics Engineering GmbH
(Headquarters)
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 77180
Fax. +43 (0)2756 7718033
office@microtronics.com
www.microtronics.com



Certified by TÜV AUSTRIA: EN ISO 9001:2015, EN ISO 14001:2015, ISO/IEC 27001:2013, EN ISO 50001:2011 for myDatanet | TÜV SÜD: ATEX Directive 2014/34/EU

© Microtronics Engineering GmbH. All rights reserved. Photos: Microtronics, shutterstock.com

Microtronics Engineering GmbH | www.microtronics.com
Hauptstrasse 7 | 3244 Ruprechtshofen | Austria | +43 2756 77180 | office@microtronics.com