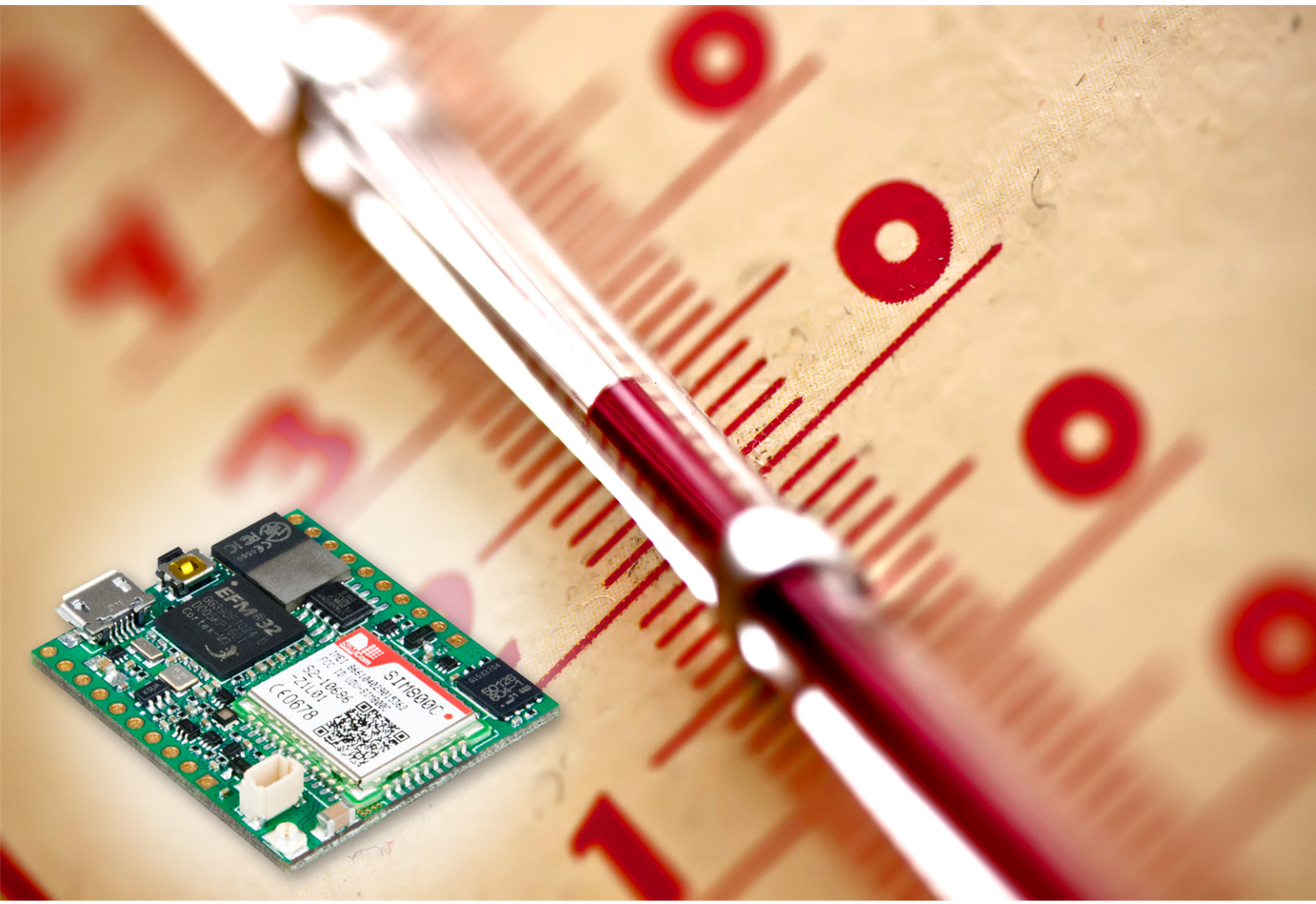
From the sensor signal
to the visualization on the web

Application for the Internet of things quickly realized



300817 | Rev.01 A

Microtronics
We live M2M

Tutorial

The following tutorial uses the example of a "Wireless Thermometer" to demonstrate how you can use the rapidM2M M3 to quickly create an application for the Internet of Things. This tutorial is split into five phases:

- Phase 1: Determining the connector
- Phase 2: Creating the application script
- Phase 3: Visualising the data on the Cloudserver
- Phase 4: Access to data and configurations via the REST-API of the Cloud server using the rapidM2M Playground
- Phase 5: Creating your own dashboard

You will learn how to specify the data structure (i.e. connector) required to exchange data between the rapidM2M M3 , Cloud server and external systems, to create an application script using the rapidM2M Toolset and to load it into the rapidM2M M3 via the USB interface. Additionally, you will learn to use the reports to display data on the Cloud server and will gain an overview of how the rapidM2M Playground can be used to read out data and configurations via the REST-API of the Cloud server. Finally, you will learn to create your own dashboard to display the data and enter a configuration parameter, while the dashboard you have created will access the corresponding information saved on the Cloud server via the REST-API.

2.1 Prerequisites

You will require the following components for this tutorial:

- PC with Microsoft operating system and up-to-date browser (recommended: Google Chrome, Mozilla Firefox)
- rapidM2M M3 (300443)
- USB cable A-connector to micro-B-connector
- rapidM2M Toolset (minimum version 1.18.4, download at: www.microtronics.com/Toolset)
- Account on the Cloud server
- Additional files for this tutorial (can be downloaded at: http://support.microtronics.at/tutorial/Tutorial_wireless_thermometer_additional_files.zip)

2.2 Detailed description

A new site to operate the wireless thermometer is created and the required connector is defined during phase 1. Application scripts for the rapidM2M M3 can have practicallyany number of configuration parameters and measurement channels. Their data type can also be selected freely within certain limits. The connector is designed to inform the Cloud server of this setup so that the measurement data and configurations can be used in conjunction with the server interface (reports, visualisations, graphics, etc.). The Cloud server must be aware of this setup to grant access to the measurement data and configurations via the API. It is recommended to start with the definition of the connector when developing a M2M/IoT application. The connection type is then changed to "online" and the record interval to 10 sec. to achieve a fast reaction time.

Phase 2 uses the example of a wireless thermometer to demonstrate how you can use the rapidM2M Toolset to create an application script for the rapidM2M M3 and load it into the device via the USB interface.

During phase 3, a report is configured to display the latest measurement values and the progress of the measurement values over the last 30 minutes via the interface of the Cloud server. In accordance with the definition of the connector determined during phase 1, the measurement values, battery voltage, USB charging voltage and temperature are provided by the application script.

Optional phase 4 illustrates how measurement data and configurations can be accessed via the REST-API of the Cloud server using the rapidM2M Playground . It is thus designed to help familiarise yourself with the provided resources and how they are handled. The first part of this phase demonstrates how the latest measurement values for the battery voltage, USB charging voltage and temperature are read out of the server via the API. In a second step, the complete configuration is initially read out and a single parameter (the connection type) is then changed from "Online" to "Wakeup".

Phase 5 explains the development of a dashboard that access the measurement data and configurations, saved on the Cloud server, via the REST-API. It includes a screen to enter the user credentials and another one to display the latest measurement values supplied by the "Wireless thermometer" application script and to enter the desired record interval.

2.3 "Wireless thermometer" script

The script records periodic measurement data and transfers it to the Cloud server. The interval for recording and transmitting measurement data can be configured. The connection type can also be configured. If the "online" connection type is selected, the measurement data is transmitted to the server as soon as the measurement data is created. The battery voltage, USB charging voltage and temperature are recorded. The current operating state is indicated by the RGB-LED. The LED lights up blue if there is an existing connection with the Cloud server. The LED flickers blue while the connection is being established. If the last connection attempt failed, the LED flashes red until the next attempt to establish a connection. However, if the last connection establishment was successful and the rapidM2M M3 is waiting for the next contact with the Cloud server (e.g. during "Interval" connection mode), then the LED is switched off. A connection to the server is either triggered automatically by the device following expiry of the transmission interval or receipt of a wakeup SMS or manually by pressing the button.

> ***Note:*** *The units specified in the following relate to the exchange of data between the rapidM2M M3 and Cloud server. When defining the connector, the "vscale" attribute can be used to adapt the units for the display on the interface of the Cloud server or output via the REST-API. The transmission interval for the display/input is thus converted from seconds to minutes, for example.*

The script generates and transmits the following measurement values:

Measurement value	Unit	Explanation
Battery voltage	mV	Voltage at the rechargeable battery connection of the rapidM2M M3 If the rapidM2M M3 is charged via USB, the measurement value corresponds to the charging voltage. Otherwise, the measurement value corresponds to the voltage of the connected rechargeable battery.
USB charging voltage	mV	Voltage at the USB connection of the rapidM2M M3
Temperature	0.1°C	Measurement value read out from the TMP112 (IC5) temperature sensor of the rapidM2M M3

The script provides the following configuration options:

Parameter	Unit	Explanation
Transmission interval	sec.	Time between transmissions
Record interval	sec.	Time between measurement data recordings
Connection type	---	0: The device connects in the transmission cycle. 1: The device connects in the transmission cycle.However, a connection can also be initiated through the server. 2: The device does not disconnect the connection and continuously transmits the measurement data.

2.4 "rapidM2M UAPI SAMPLE" dashboard

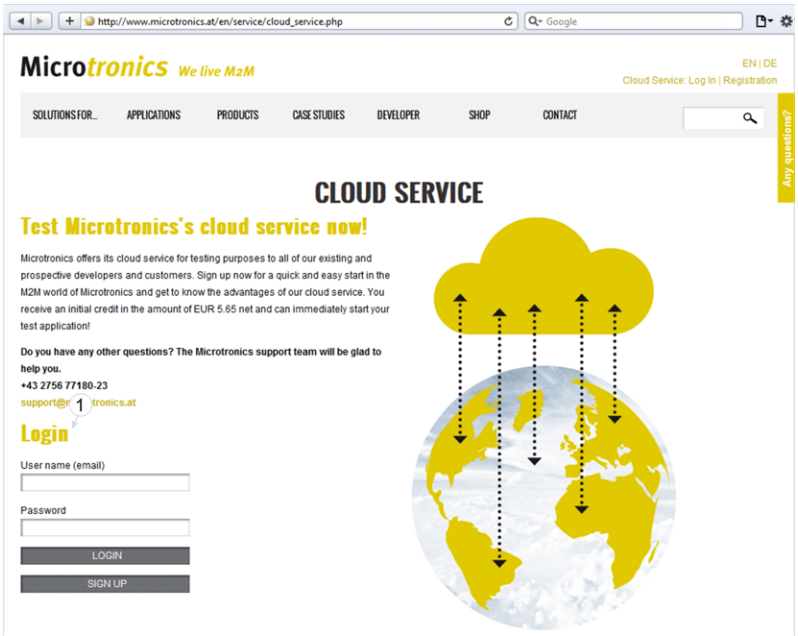
The dashboard has two views that are referred to as the "login page" and "content page" in the following. The login page, that enables the entry of the user credentials, is initially displayed after opening the dashboard. If valid login data is entered, this switches to the content page. In addition to issuing the latest measurement values for the battery voltage, USB charging voltage and temperature provided by the application script, the record interval can also be entered via the

content page. The site, for which the information is displayed, is selected automatically. The first site of the first customer, for whom the user has access rights, is selected. The selection is displayed on the content page above the area for the measurement values. The measurement values themselves are updated at one second intervals. The logout button can be used to switch back to the login Page, which also deactivates the measurement values from being updated. When switching back to the login page, the user credentials entered in the input fields are not deleted. This has proven particularly convenient for the test phase during which you frequently switch between the two views. Although this feature should not be implemented in productive dashboards in this form under any circumstances. The dashboard also has an area to display status information (e.g. error codes) that is available in both views. Screenshots of both the views and a detailed explanation of the individual elements is provided in chapter "Phase 5: Creating your own dashboard" auf Seite 37.

Step-by-step instructions

3.1 Phase 1: Determining the connector

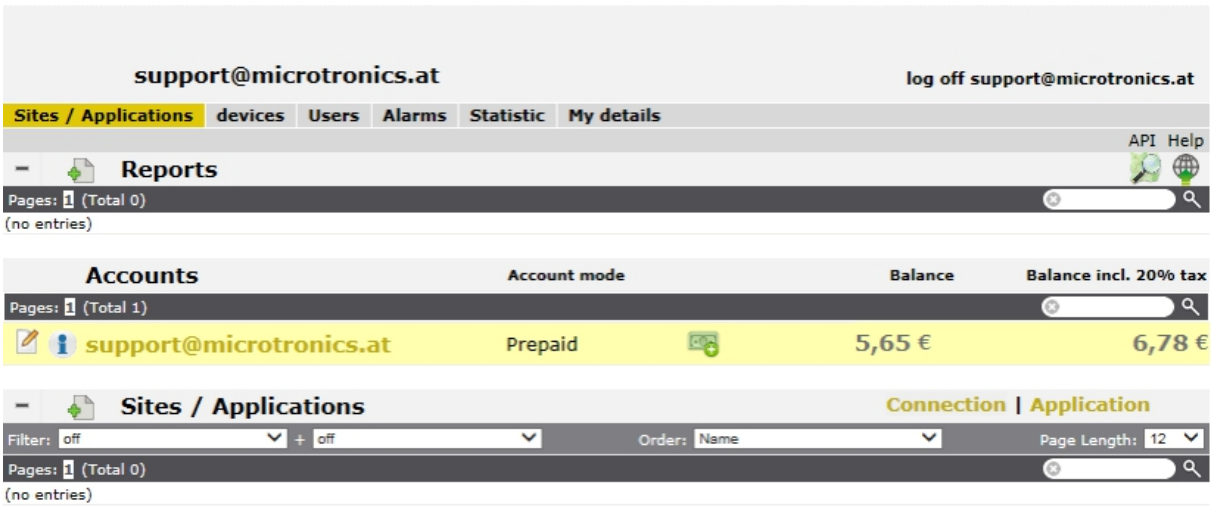
1. If you do not already have an account on the Cloud server, you can create a free account via the Microtronics Engineering GmbH (cloud.microtronics.com) homepage.
2. Log into the Cloud server (cloud.microtronics.com). Your user name is the e-mail address that you specified when creating the account.



Overview of the homepage

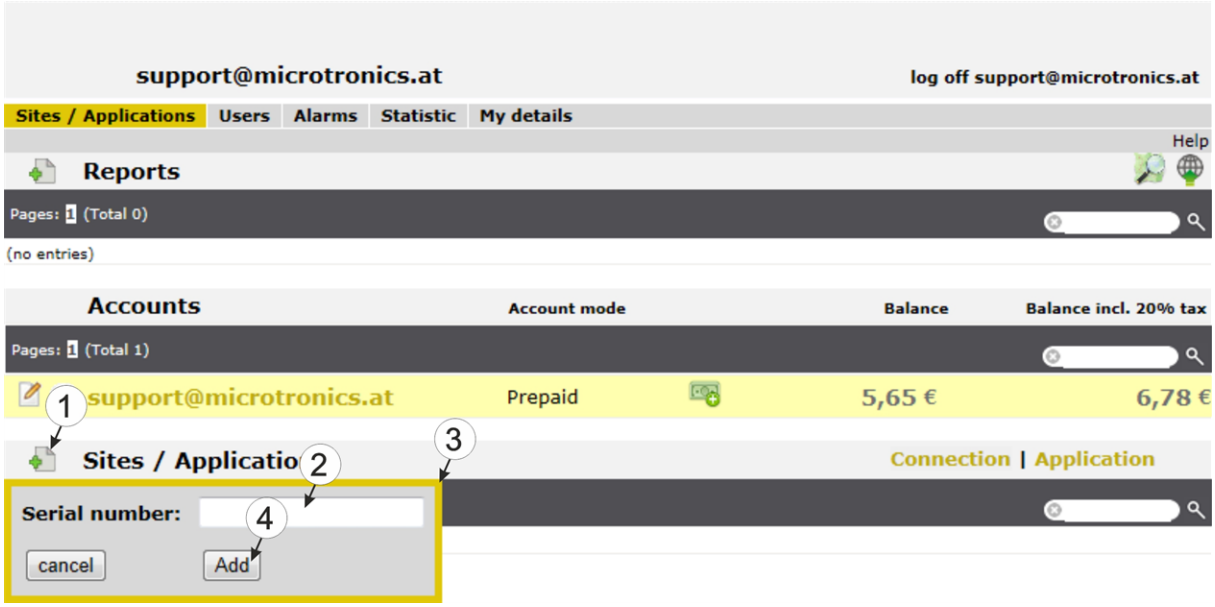
1	Access to the Cloud server	

3. You are now in the "Sites/applications" area of the Cloud server.



"Sites/applications" area where all of the available sites are displayed.

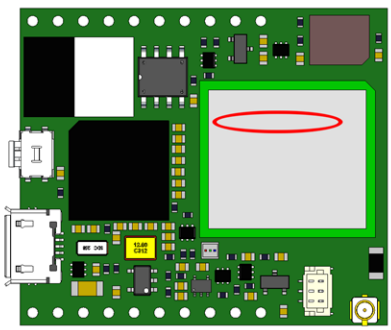
4. Create a site.



Creating the site

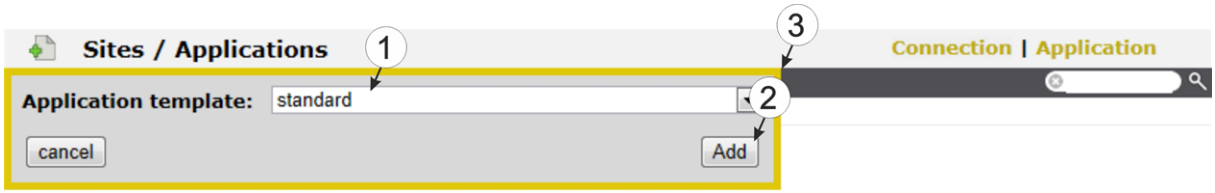
1	"Add new site/application" symbol	3	Input window for creating a new site
2	Field for entering the serial number	4	"Add" button

The serial number corresponds to the IMEI of the rapidM2M M3 supplemented by a final zero. The IMEI is located on the modem of the rapidM2M M3 .



rapidM2M M3

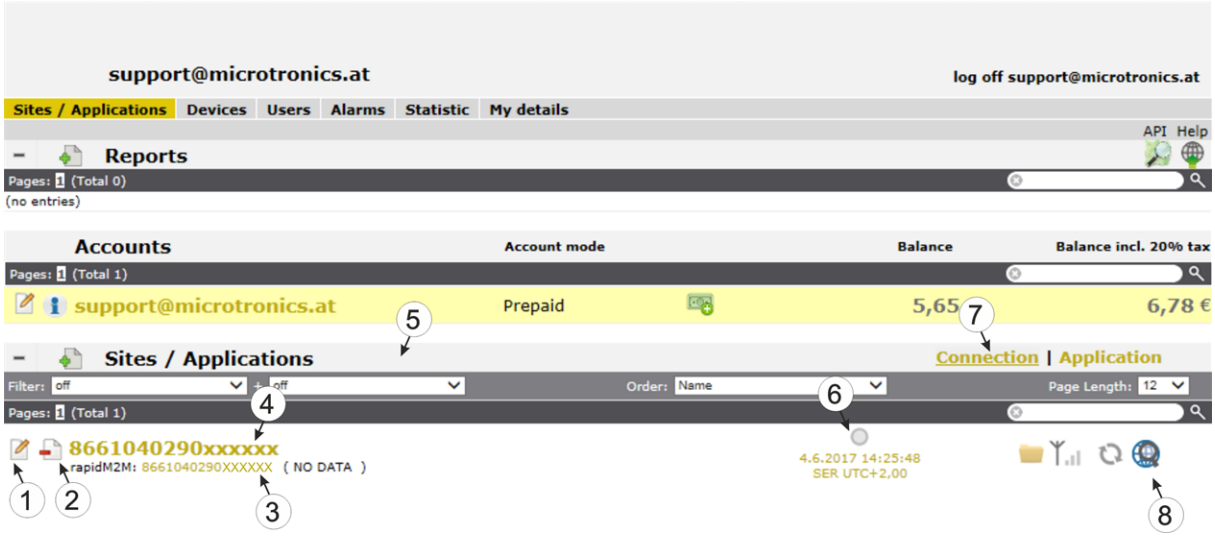
Select "**Standard**" as the application template.



Selecting the application template

1	Dropdown list of available application templates	3	Input window for selecting the application template
2	"Add" button		

5. The newly added site is now displayed in the list of sites.

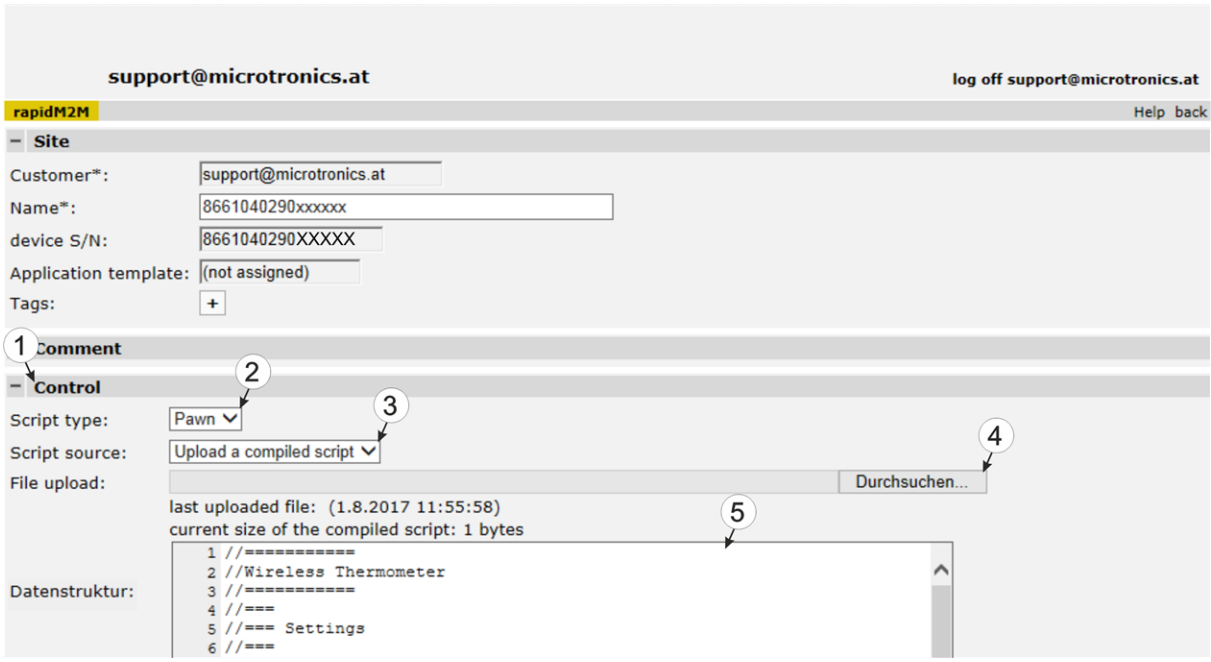


"Sites/applications" area where all of the available sites are displayed.

1	Opens the input screen for configuring the site
2	Deletes the site/application.
3	Serial number of the device linked to the site. The input screen for configuring the device is opened by clicking on the serial number .
4	Name of the site/application. The input screen for configuring the site/application is opened by clicking on the name .
5	List of sites/applications
6	Status of the measurement instrument. In addition to the status symbol, additional symbols are displayed depending on the current operating state.
7	Tabs to switch between the display of the connection-specific and application-specific control elements
8	Connection or application-specific control elements of the site/application

6. Then click on the name of the site in the list of sites/applications to open the input screen for configuring the site.

7. In the input screen for configuring the site, click on "Control" to display the input fields for uploading a compiled script and for configuring the data structure (i.e. the connector). Set the parameters as follows:



Input screen for configuring the site/application, "Control" section

1	"Control" configuration section
2	Select the script type Select "Pawn" here. ***Note:*** *The "Script source" parameter is only displayed here following this selection.*
3	Specification of whether the script should be edited directly on the Cloud server or whether a previously compiled script should be used. Select "Upload a compiled script" here. ***Note:*** *The "Upload file" parameter is only displayed following this selection.*
4	Button to open the dialog to select the script binary file (*.amx) to be uploaded Select the "Wireless_Thermometer.amx" file here, if you want to skip the step-by-step creation of the application script during phase 2. It is included in the package of additional files for this tutorial.
5	Input field to configure the connector that enables the data generated and configurations provided by the script to be used in connection with the interface of the Cloud server Successively copy the blocks, described in the following three steps, into this field. If you would like to skip this step-by-step creation of the connector, you can copy the entire content of the "Datastructure.txt" file into this field and continue the tutorial at step 11. The "Datastructure.txt" file is included in the package of additional files for this tutorial.

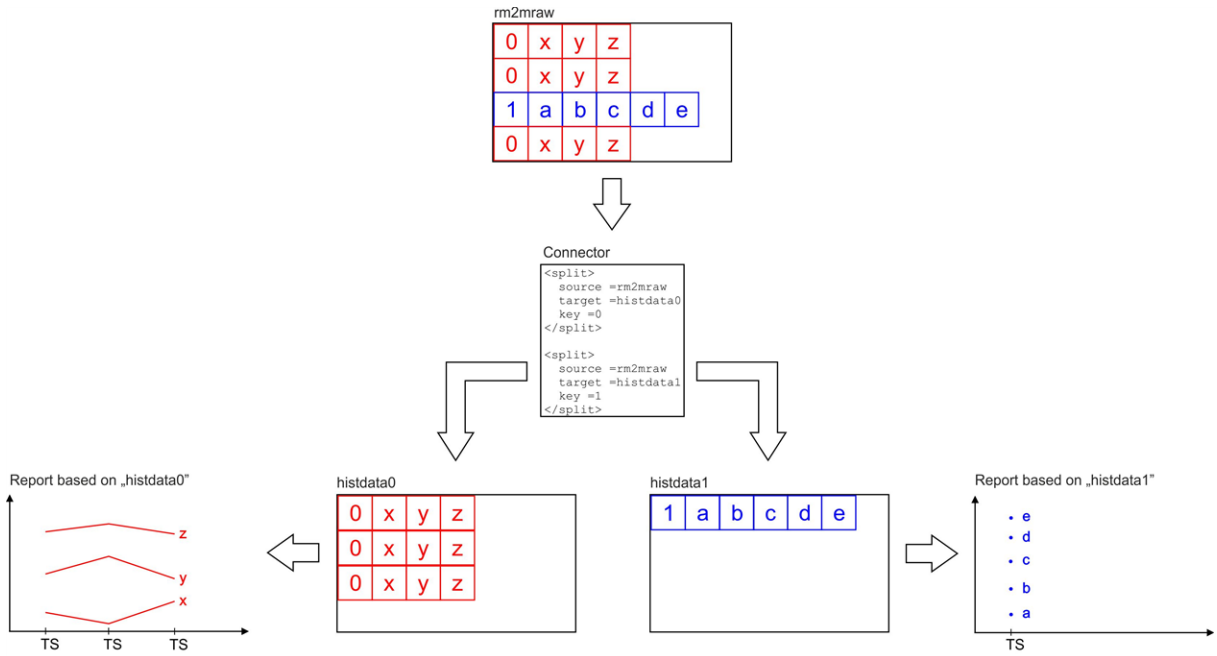
8. You must first decide which configuration parameters are required for the desired function. In this particular case, this includes a parameter to select the transmission interval, one to select the record interval and one to select the connection type (interval, wakeup or online). The following table contains additional requirements to define the structure of the configuration parameters:

Parameter	Data type	Unit/selection options	Decimal places	Default value
Transmission interval	32Bit unsigned	min.	None	60 min.
Record interval	32Bit unsigned	sec.	None	60 sec.
Connection type	8Bit unsigned	0 = Interval 1 = Wakeup 2 = Online	None	Interval

Due to the decision to use configuration block 0, the configuration has the following data structure:

```
<table>
    //Configuration block 0 should be used
    name  = config0
    //Title of the config. section displayed on the server
    title = Configuration
    <field>
        //Parameter 0 should be used for the transmission interval
        name       = field0
        //Alternative field name that can be used by the REST-API
        alias      = RecordInterval
        //Title of the parameter displayed on the server
        title      = Record Interval
        //Position in config block 0 where the parameter is saved
        byteofs    = 0
        //No decimal places
        decpl      = 0
        //Data type: 32Bit unsigned
        type       = u32
        //Unit to be used to enter the value for the parameter
        units      = sec
        //Lowest valid value for the parameter
        min        = 10
        //Default value is 60 sec.
        default    = 60
    </field>
    <field>
        //Parameter 1 should be used for the record interval
        name       = field1
        alias      = TransmissionInterval
        title      = Transmission Interval
        byteofs    = 4
        decpl      = 0
        type       = u32
        units      = min
        min        = 10
        default    = 60
        //Conversion factor sec.->min. (sec. used internally)
        vscale     = 0.01666666667
    </field>
    <field>
        //Parameter 2 should be used for the connection type
        name       = field2
        alias      = TransmissionMode
        title      = Transmission Mode
        byteofs    = 8
        decpl      = 0
        type       = u8
        default    = 0
        //Selection options for the dropdown list
        editmask   = 0=Interval;1=Wakeup;2=Online
    </field>
</table>
```

9. The next step is to specify how many and which measurement data channels of the Cloud server should be used. The rapidM2M M3 can save data records of different lengths and structures in its internal memory. The Cloud server, on the other hand, always requires data records with the same structure. The received data records must therefore be distributed to the individual measurement data channels of the Cloud server based on their structure. This is completed using the split tag.



Graphic display of the functionality of the split tags

In the current example, the data records will always have the same length and structure, which means that only one of the measurement data channels of the Cloud server is required. In conjunction with the decision to use measurement data channel 0, the following structure for the split tag is determined:

```
<split>
    //Raw data channel (always "rm2mraw" for the rapidM2M M3 ) from which the data records should be
    //copied
    source = rm2mraw
    //Measurement data channel to which the data record should be copied
    target = histdata0
    //If the first byte of the data record corresponds to this value, it is copied to the specified
    //measurement data channel.
    key    = 00
</split>
```

10. Finally, it must be defined which measurement values are recorded by the application that has to be designed. In this example, this refers to the battery voltage, USB charging voltage and temperature. The following table contains additional requirements to define the structure of the measurement values:

Measurement value	Data type	Unit	Decimal places
Battery voltage	16Bit signed	V	2
USB charging voltage	16Bit signed	V	2
Temperature	16Bit signed	°C	1

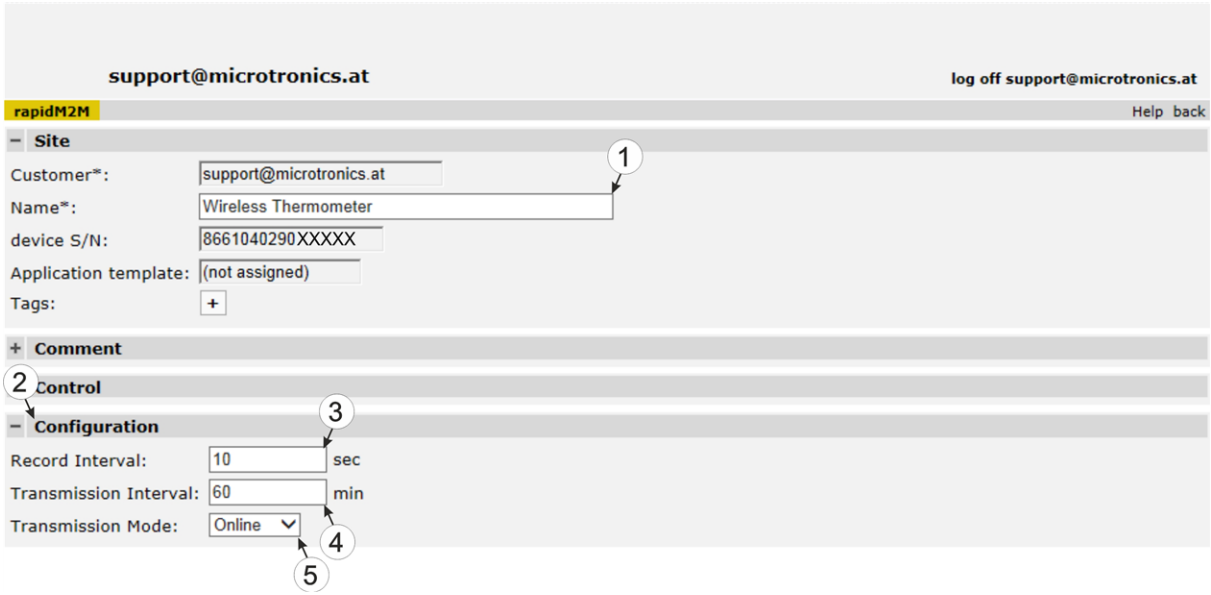
Due to the decision, made in the previous step, to use measurement data channel 0, the measurement values have the following data structure:

```
<table>
    //Measurement data channel 0 should be used.
    name = histdata0
    <field>
        //Data field 0 should be used for the battery voltage.
        name    = ch0
        //Alternative field name that can be used by the REST-API
        alias   = VoltageBattery
        //Title of the data field displayed on the server
        title   = Battery Voltage
        //Position in measurement data channel 0 where the data field is saved
        //Note: The key for the split tag is located at position "0"
        byteofs = 1
        //The measurement value should be rounded off to two decimal places.
        decpl   = 2
        //Data type: 16Bit signed
        type    = s16
        //Conversion factor mV->V (V is used internally)
        vscale  = 0.001
        //Measurement value unit displayed on the server
        units   = V
    </field>
    <field>
        //Data field 1 should be used for the USB charging voltage.
        name    = ch1
        alias   = VoltageUSB
        title   = USB Voltage
        byteofs = 3
        decpl   = 2
        type    = s16
        vscale  = 0.001
        units   = V
    </field>
    <field>
        //Data field 2 should be used for the temperature.
        name    = ch2
        alias   = Temperature
        title   = Temperature
        byteofs = 5
        decpl   = 1
        type    = s16
        vscale  = 0.1
        units   = °C
    </field>
</table>
```

11. Exit the input screen for configuring the site by clicking the "Save" button. Following the saving process, the configuration parameters and measurement channels specified by the connector can be used on the interface of the Cloud server.

12. Open the input screen to configure the site again by clicking on the name of the site in the list of sites/applications.

13. In the input screen for configuring the site, click on "Configuration" to display the input fields for the record interval, transmission interval and connection type. Set the parameters as follows. Change the name of the site at the same time to avoid any subsequent confusion between the site name and serial number. Then exit the input screen by clicking the "Save" button.

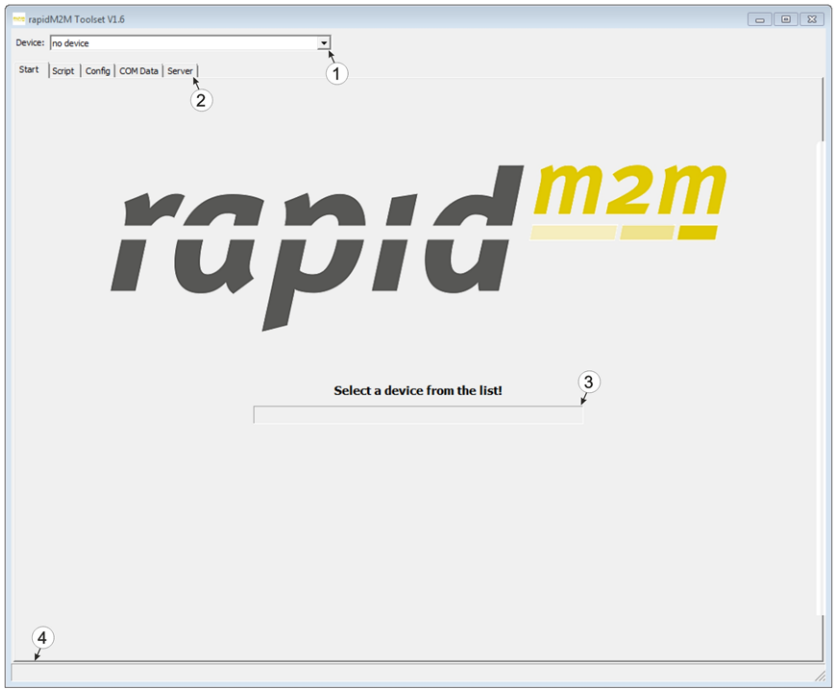


Input screen for configuring the site/application, "Configuration" section

1	Name of the site (freely selectable) Select "Wireless Thermometer" here
2	"Configuration" configuration section
3	Time between measurement data recordings Select "10 sec." here.
4	Time between transmissions (only applies to the "Interval" and "Wakeup" connection types) Select "60 min." here.
5	Selection of the connection type Select "online" here. The device does not disconnect the connection and continuously transmits the measurement data.

3.2 Phase 2: Creating the application script

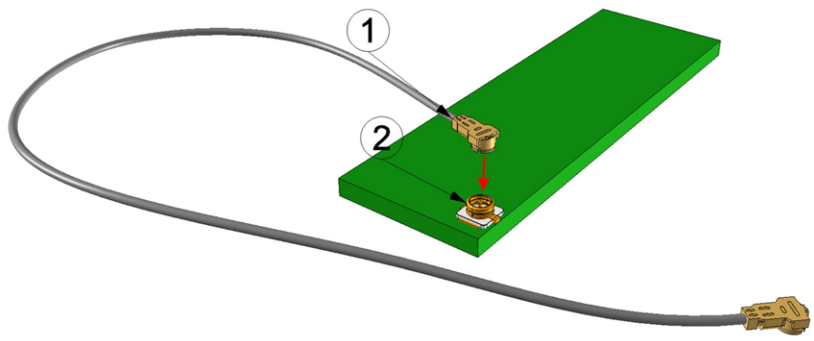
1. Install the rapidM2M Toolset . It can be downloaded free of charge from the following website: www.microtronics.com/Toolset.
2. Start the rapidM2M Toolset .



"Start" tab (no rapidM2M M3 selected)

1	Selection of the rapidM2M M3	3	Progress bar for establishing the USB connection to the rapidM2M M3
2	Selection of the tab for the various functions of the rapidM2M Toolset , that are even available without the connected rapidM2M M3	4	Indication of the communication status on the USB interface to the rapidM2M M3

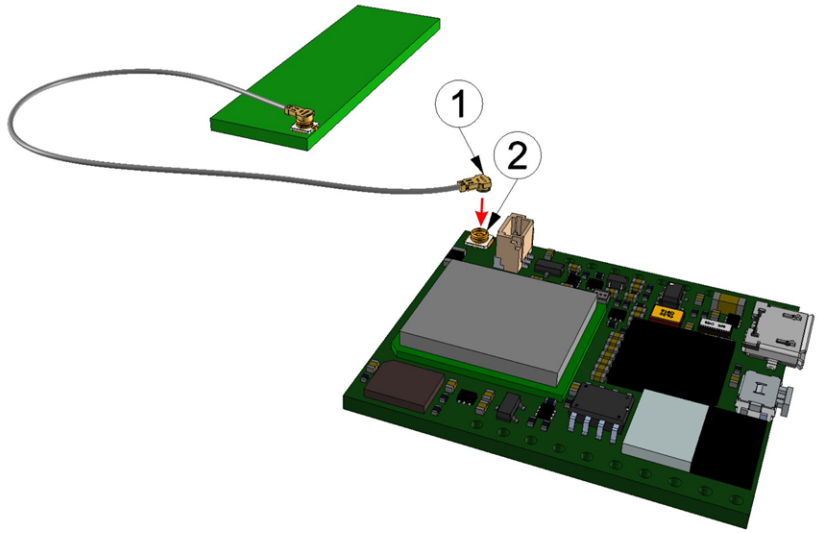
3. Connect the U.FL antenna cable to the U.FL antenna connector of the PCB antenna.



Connecting the antenna cable to the antenna

1	U.FL antenna cable	2	U.FL antenna connector of the PCB antenna

4. Connect the U.FL antenna cable to the U.FL antenna connector of the rapidM2M M3

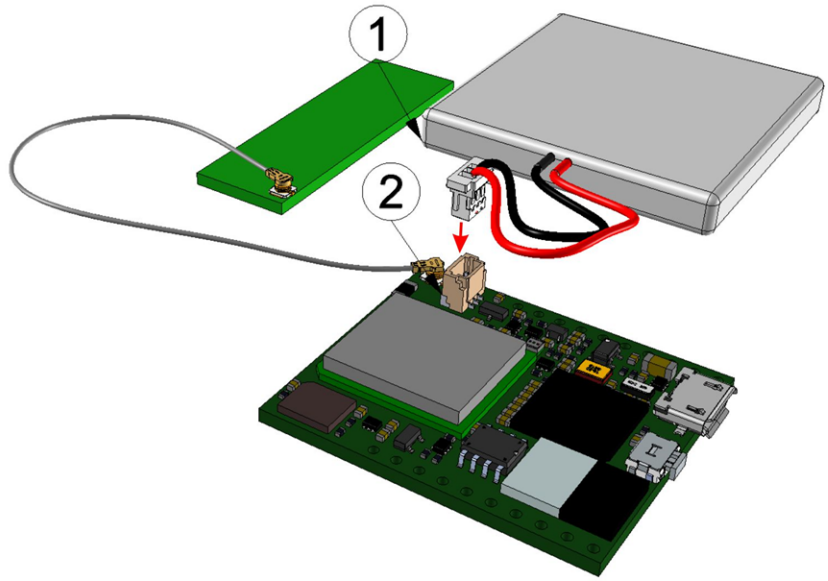


Connecting the antenna cable to the rapidM2M M3

1	U.FL antenna cable	**2**	U.FL antenna connector of the rapidM2M M3

5. Connect the rechargeable battery to the rechargeable battery connection of the rapidM2M M3 . If a script has not been loaded in the rapidM2M M3 yet, the green LED then starts to flicker to indicate that a connection is being established.

> ***Note:*** *Before connecting the rechargeable battery, ensure that no objects made of bare metal are in the vicinity of the rapidM2M M3 as these could cause a short circuit.*

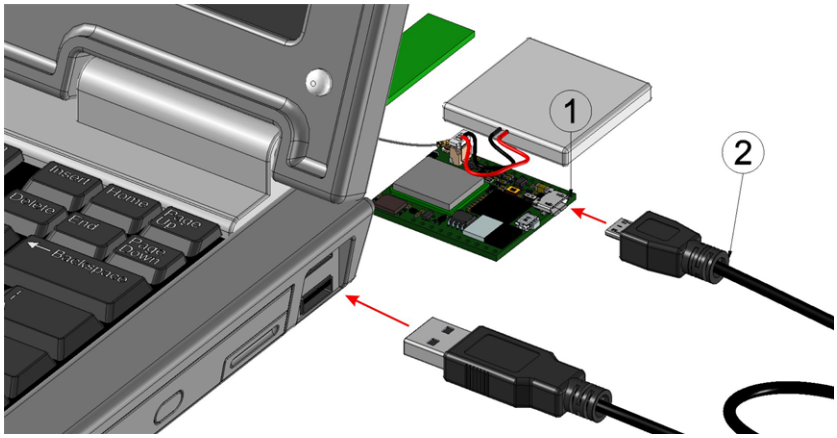


Connecting the rechargeable battery to the rapidM2M M3

1	Li-Po rechargeable battery	**2**	Rechargeable battery connector of the rapidM2M M3

6. Connect the rapidM2M M3 to your PC using a USB cable (A-connector to micro-B-connector) and wait until the automatic installation of the drivers required for the rapidM2M M3 has completed. The Li-Po rechargeable battery of the rapidM2M M3 is also charged via the USB interface.

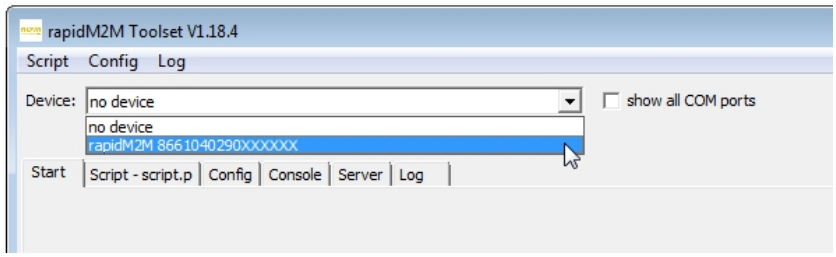
   Note: *The Li-Po rechargeable battery should be charged approx. 12 hrs before initial start-up.*



Connecting therapidM2M M3 to the PC

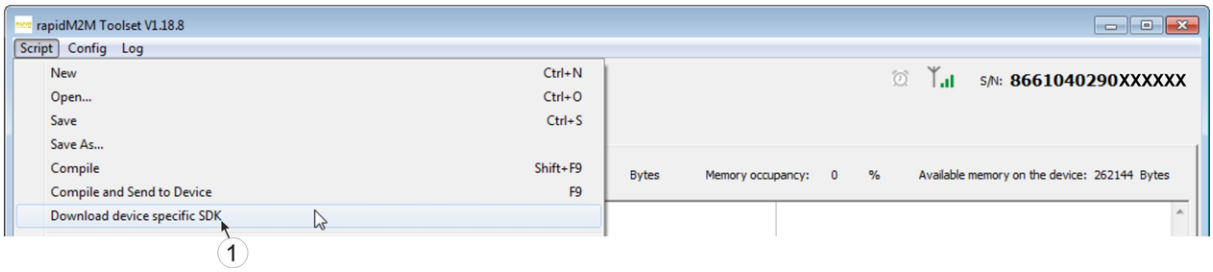
1	Micro-USB connector of the rapidM2M M3	2	USB cable (A-connector to micro-B-connector)

7. Now switch to the rapidM2M Toolset and select your rapidM2M M3 based on the serial number from the list of found devices.



Device selection

8. Load the specific extensions for the rapidM2M M3 via the "Script -› Download device specific SDK" menu item.

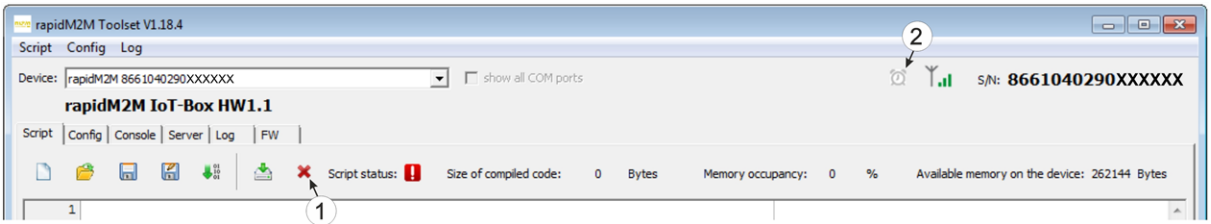


Downloading device-specific SDK

1	Downloading or updating the specific extensions for the device that is currently connected to the PC

9. Delete the script installed in the rapidM2M M3 (symbol). After briefly flashing (connection is being established), the display of the GSM connection status changes to green (connected) and then to the symbol (offline, wakeup possible via Cloud server).

> ***Note:*** *If you have already loaded the "Wireless_Thermometer.amx" script binary on the Cloud server during phase 1, i.e. you have decided to skip the step-by-step creation of the application script, the GSM connection status switches from green (connected) to grey (offline). In this case, continue the tutorial at step 22.*



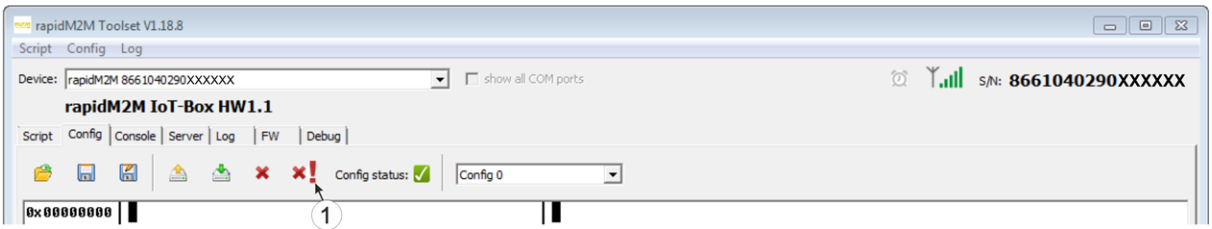
Deleting the script installed in the rapidM2M M3

1	Deletes the installed script from therapidM2M M3
2	Display of the GSM connection status. A connection establishment can be started by clicking on this symbol.

GSM connection status

Status symbol	
	Offline
	Offline, wakeup possible via the Cloud server
, flashing	Connection is being established
	Connected
	Last connection attempt failed
	Last connection attempt failed, wakeup possible via the Cloud server

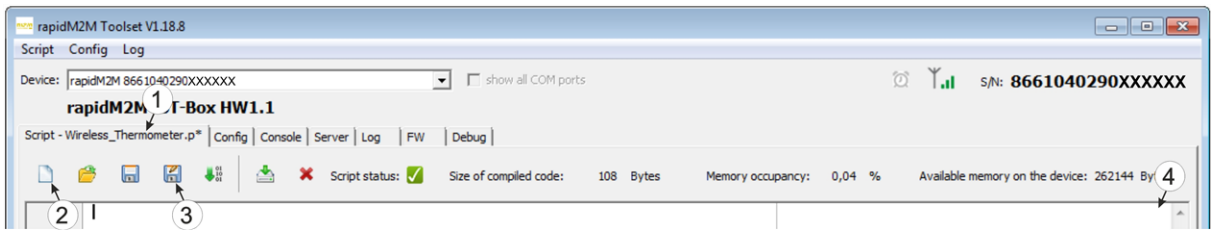
10. Now select the "Config" tab and also delete the content of the configuration blocks of the rapidM2M M3 (symbol).



Deleting the content of the configuration blocks

1	Deletes the content of all the configuration blocks of the rapidM2M M3

11. Switch back to the "Script" tab, create a new empty script (symbol) and save it on your hard drive (symbol).



Creating a new script

1	File name of the opened script. If the script was changed but has not been saved yet, this is indicated by "*" after the file name.
2	Creates a new empty script
3	Opens the dialogue to save the script under a different file name
4	Script editor Successively copy the blocks, described in the following nine steps, into this field. If you would like to skip this step-by-step creation of the application script, you can copy the entire content of the "Wireless_ Thermometer.p" file into this field and continue the tutorial at step 21. The "Wireless_Thermometer.p" file is included in the package of additional files for this tutorial.

12. First of all, you must specify the path for the specific include file for the rapidM2M M3 and the function prototypes of the callbacks called up for corresponding events by the firmware.

```
/* Path for hardware-specific include file */
#include ".\rapidM2M IoT-Box\iotbox"

/* Forward declarations of the public functions */
forward public Timer1s();                    // called up 1x per sec. for the program sequence
forward public ReadConfig(cfg);              // called up when one of the config blocks is changed
forward public KeyChanged(iKeyState);        // called up when the button is pressed or released
```

13. A few constants and global variables for the program sequence must be defined in the next step.

```
/* Standard values to initialise the configuration */
const
{
  ITV_RECORD         = 1 * 60,               // Record interval [sec.], default 1 min
  ITV_TRANSMISSION   = 1 * 60 * 60,          // Transmission interval [sec.], default 60 min
  TXMODE             = RM2M_TXMODE_TRIG,     // Connection type, default "Interval"

  DEFAULT_COLOR      = 0x00008ECF,           // LED colour
}

/* Size and index specifications for the configuration blocks and measurement data block */
const
{
  CFG_BASIC_INDEX = 0,                       // Config block 0 contains the basic config
  CFG_BASIC_SIZE = 9,                        // Record interval (u32) + transmission interval (u32) +
                                             // Connection type (u8)

  HISTDATA_SIZE = 3 * 2 + 1,                 // 3 channels (s16) + "Split tag" (u8)
}

/* Global variables to store the current configuration */
static iRecItv;                              // current record interval [sec.]
static iTxItv;                               // current transmission interval [sec.]
static iTxMode;                              // current connection type (0 = interval,
                                             // 1 = wakeup, 2 = online)

/* Global variables for the remaining time until certain actions are triggered */
static iRecTimer;                            // sec. until the next recording
static iTxTimer;                             // sec. until the next transmission
```

14. The main function can now be created.

```
main()
{
  /* Temporary memory for the index of a public function and the return value of a function            */

  new iIdx, iResult;

  /* Initialisation of the button -> Evaluation by the script activated
     - Determining the function index that should be called up when pressing or releasing the button
     - Transferring the index to the system and inform it that the button is controlled by the script
     - Index and return value of the init function are issued by the console                           */
  iIdx = funcidx("KeyChanged");
  iResult = Switch_Init(SWITCH_MODE_SCRIPT, iIdx);
  printf("Switch_Init(%d) = %d\r\n", iIdx, iResult);

  /* Initialisation of the LED -> Control via script activated */
  LED_Init(LED_MODE_SCRIPT);

  /* Initialisation of a 1 sec. timer used for the general program sequence
     - Determining the function index that should be executed 1x per sec.
     - Transferring the index to the system
     - Index and function return value used to generate the timer are issued via the console           */
  iIdx = funcidx("Timer1s");
  iResult = rM2M_TimerAdd(iIdx);
  printf("rM2M_TimerAdd(%d) = %d\r\n", iIdx, iResult);

  /* Specification of the function that should be called up when a config block is changed
     - Determining the function index called up when changing one of the config blocks
     - Transferring the index to the system
     - Index and init function return values are issued via the console                                */
  iIdx = funcidx("ReadConfig");
  iResult = rM2M_CfgOnChg(iIdx);
  printf("rM2M_CfgOnChg(%d) = %d\r\n", iIdx, iResult);

  /* Read out the basic configuration. If config block 0 is still empty (first program start), the
     basic configuration is initialised with the standard values.
     */
  ReadConfig(CFG_BASIC_INDEX);

  /* Setting the counters to 0 immediately triggers a transmission and a recording.
     You could also refrain from setting it to 0, as all variables in PAWN are initialised with 0
     when they are created. However, it was completed at this point for the purpose of a better
     understanding.                                                                                   */
  iTxTimer  = 0;
  iRecTimer = 0;

  /* Setting the connection type */
  rM2M_TxSetMode(iTxMode);
}
```

15. The function that is called up once per second by the system and is thus essential for the program sequence, must then be created.

```
/* 1sec. timer is used for the general program sequence */
public Timer1s()
{
  Handle_Led();                          // Control of the LED
  Handle_Transmission();                 // Control of the transmission
  Handle_Record();                       // Control of the record
}
```

16. The function, that controls the LED, must be generated now.

```
/* Function to control the LED */
Handle_Led()
{
  new iTxStatus;                          // Temporary memory for the connection status

  iTxStatus = rM2M_TxGetStatus();         // Read current connection status from the system

  if(iTxStatus & RM2M_TX_ACTIVE)          // If a GPRS connection is currently established ->
  {
    Led_Off();                            // Switch off LED
    Led_On(DEFAULT_COLOR);                // Switch on LED (default LED colour)
  }
  /* If a connection attempt is currently being executed or the delay until the retry is in progress -> */
  else if(iTxStatus & (RM2M_TX_STARTED|RM2M_TX_RETRY))
  {
    Led_Off();                            // Switch off LED
    Led_Flicker(0, DEFAULT_COLOR);        // LED flickers continuously (default LED colour)
  }
  else if( iTxStatus & RM2M_TX_FAILED)    // If the last connection attempt failed ->
  {
    Led_Off();                            // Switch off LED
    Led_Blink(0, 0x00FF0000);             // LED flashes red continuously
  }
  else                                    // Otherwise ->
    Led_Off();                            // Switch off LED
}
```

17. The next function that needs to be generated controls the handling of the transmission interval.

```
/* Function to generate the transmission interval */
Handle_Transmission()
{
  iTxTimer--;                             // Counter counting down the sec. to the next transmission
  if(iTxTimer <= 0)                       // When the counter has expired ->
  {
    rM2M_TxStart();                       // Establish a connection to the server
    iTxTimer = iTxItv;                    // Reset counter var. to current transmission interval [sec.]
  }
}
```

18. The function, that records the data, must then be designed.

```
/* Function for recording the data */
Handle_Record()
{
  /* Temporary memory in which the data record to be saved is compiled. */
  new aRecData{HISTDATA_SIZE};

  iRecTimer--;                              // Decrementing the Counter counting down the seconds to the
                                            // next recording
  if(iRecTimer <= 0)                        // When the counter has expired ->
  {
    new aSysValues[TIoTbox_SysValue];       // Temporary memory for the int. measurement values (VBat,
                                            // VUsb, temp)
    IoTbox_GetSysValues(aSysValues);        // Read out the current int. measurement values (VBat, VUsb,
                                            // temp)

    /* Compile the data record to be saved in the "aRecData" temporary memory
       - The first byte (position 0 in the "aRecData" array) is set to 0 so that the server copies
         the data record into measurement data channel 0 upon receipt, as specified during the design
         of the connector
       - The battery voltage (VBat) is copied to position 1-2. Data type: s16
       - The USB charging voltage (VUsb) is copied to position 3-4. Data type: s16
       - The temperature (Temp) is copied to position 5-6. Data type: s16                              */
    aRecData{0} = 0;                        // "Split tag"
    rM2M_Pack(aRecData,  1,   aSysValues.VBat, RM2M_PACK_BE + RM2M_PACK_S16);
    rM2M_Pack(aRecData,  3,   aSysValues.VUsb, RM2M_PACK_BE + RM2M_PACK_S16);
    rM2M_Pack(aRecData,  5,   aSysValues.Temp, RM2M_PACK_BE + RM2M_PACK_S16);

    /* Transfer compounded data record to the system to be recorded */
    rM2M_RecData(0, aRecData, HISTDATA_SIZE);

    iRecTimer = iRecItv;                    // Reset counter variable to current record interval

    /* Issue current measurement values via the console */
    printf("Vb:%d Vu:%d Ti:%d\r\n",aSysValues.VBat, aSysValues.VUsb, aSysValues.Temp);
  }
}
```

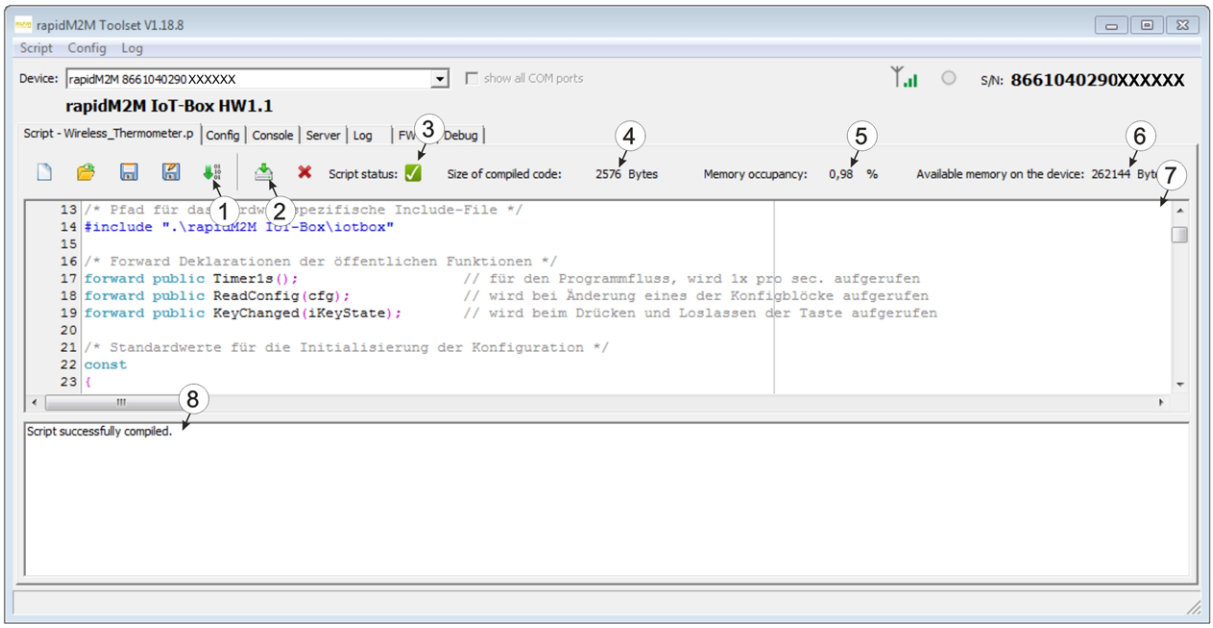
19. The function, that is called up by the system by pressing or releasing the button and via which a transmission can be triggered when the button is pressed, is created next.

```
/* Function that enables a transmission to be triggered when the button is pressed */
public KeyChanged(iKeyState)
{
  printf("K:%d\r\n", iKeyState);            // Issue action via the console (0=release, 1=press)

  if(!iKeyState)                            // If the button has been released->
  {
    /* Set count variable for seconds until the next transmission to 0. This ensures that a transmission
       is triggered the next time the "Handle_Transmission" function is called up by the "Timer1s"
       function.                                                                                      */
    iTxTimer = 0;
  }
}
```

20. Finally, it is necessary to generate the function that is called up by the system every time one of the config blocks is changed and thus makes it possible to react to a changed configuration received from the server.

```
/* Function that makes it possible to react to a changed configuration received from the server */
public ReadConfig(cfg)
{
  // If the changed configuration is the basic config -> */
  if(cfg == CFG_BASIC_INDEX)
  {
    new aData{CFG_BASIC_SIZE};              // Temporary memory for the basic config read from the system
    new iSize;                              // Temporary memory for the size of the basic config in bytes
    new iTmp;                               // Temporary memory for a basic config parameter

    /* Read basic config from the system and copy to the temporary memory. The
       number of the config block and return value of the read function (number of bytes or error code)
       is then issued via the console                                                                    */
    iSize = rM2M_CfgRead(cfg, 0, aData, CFG_BASIC_SIZE);
    printf("Cfg %d size = %d\r\n", cfg, iSize);


    /* If the number of read bytes is lower than the size of the basic config ->
      Info: Error codes are negative                                                                     */
    if(iSize < CFG_BASIC_SIZE)
    {
      /* The following block initially copies the default values to the temporary memory for the read
         basic config and then sets the temporary memory for the size of the basic config to the current
         size. This ensures that the following "IF" is executed during re-initialisation and when
         changes are received. If special actions have to be executed for individual parameters,
         it is thus not necessary to implement these separately for both cases.                          */
      iTmp = ITV_RECORD;
      rM2M_Pack(aData, 0, iTmp,  RM2M_PACK_BE + RM2M_PACK_U32);
      iTmp = ITV_TRANSMISSION;
      rM2M_Pack(aData, 4, iTmp,   RM2M_PACK_BE + RM2M_PACK_U32);
      iTmp = TXMODE;
      rM2M_Pack(aData, 8, iTmp,  RM2M_PACK_BE + RM2M_PACK_U8);
      iSize = CFG_BASIC_SIZE;
      print("created new Config #0\r\n");
    }


    /* If the number of read bytes at least equates to the size of the basic config -> */
    if(iSize >= CFG_BASIC_SIZE)
    {
      /* Copy record interval (u32, Big Endian) at position 0-3 from the temporary memory for the read
         basic config into the temporary memory for a parameter                                          */

      rM2M_Pack(aData, 0, iTmp,  RM2M_PACK_BE + RM2M_PACK_U32 + RM2M_PACK_GET);
      if(iTmp != iRecItv) // If the received value does not correspond to that of the global variables ->
      {
        printf("iRecItv changed to %d s\r\n", iTmp);  // Issue received value via the console
        iRecItv = iTmp;                               // Copy received value in to global variable
      }

      /* Copy transmission interval (u32, Big Endian) at position 4-7 from the temporary memory for the
         read basic config into the temporary memory for a parameter                                     */
      rM2M_Pack(aData, 4, iTmp,  RM2M_PACK_BE + RM2M_PACK_U32 + RM2M_PACK_GET);
      if(iTmp != iTxItv)  // If the received value does not correspond to that of the global variables ->
      {
        printf("iTxItv changed to %d s\r\n", iTmp);   // Issue received value via the console
        iTxItv = iTmp;                                // Copy received value in to global variable
      }

      /* Copy connection type (u8) at position 8 from the temporary memory for the read basic config into
         the temporary memory for a parameter                                                            */
      rM2M_Pack(aData, 8, iTmp,  RM2M_PACK_BE + RM2M_PACK_U8 + RM2M_PACK_GET);
      if(iTmp != iTxMode)//  If the received value does not correspond to that of the global variables ->
      {
        rM2M_TxSetMode(iTmp);                         // Set connection type to the received value
        printf("iTxMode changed to %d\r\n", iTmp);    // Issue received value via the console
        iTxMode = iTmp;                               // Copy received value into the global variable
      }
    }
  }
}
```

21. Compile and load the script you have just created via the symbol into the rapidM2M M3 . The status LED of the rapidM2M M3 starts to flicker blue following the script download to indicate that a connection is being established.



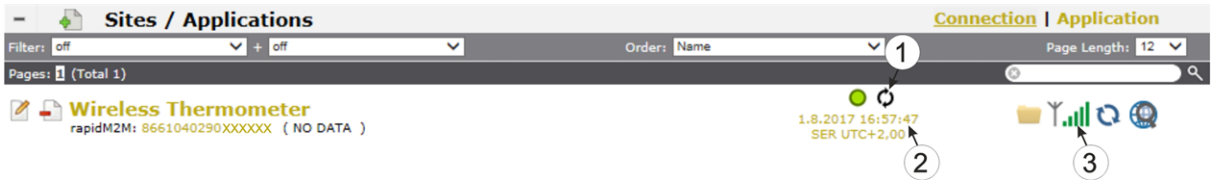
"Script" tab

1	Compiles the opened script. It is also automatically saved during this process.
2	Transfers the opened script to the module/device. During this process, the script is initially saved and then compiled. The compiled PAWN binary (*.amx) is saved in the same folder as the script.
3	Status of the script transfer
4	Size of the compiled code
5	Specifies the assignment of the available memory as a %
6	Specifies the size of the available memory on the rapidM2M M3 (specification of the compressed size)
7	Script editor
8	Debug output of the script compiler. If you double click on an entry in the debug output, the cursor will jump to the relevant line in the script editor.

Script status

Symbol	Description
	Script was successfully loaded into the rapidM2M M3 .
	Script is currently being transmitted to the rapidM2M M3 .
	The time stamp of the compiled script saved in the rapidM2M M3 does not match the compiled script of the rapidM2M Toolset .

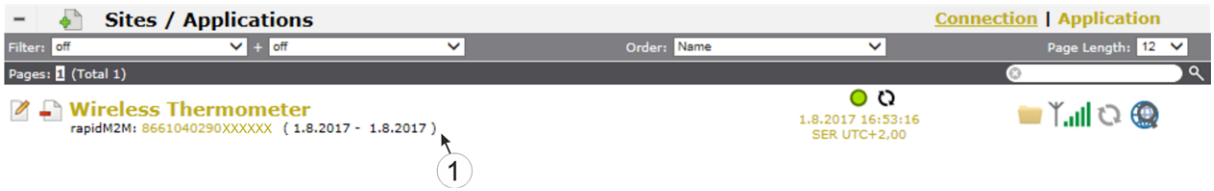
22. Switch to the Cloud server and wait until the list of sites/applications indicates that the rapidM2M M3 is connected to the Cloud server (rotating arrows). This can take up to two minutes.



List of sites/applications

1	Indicates that the rapidM2M M3 is connected to the server and is transmitting data. This symbol is hidden when the connection is disconnected.
2	Information regarding the time of the communication between the measurement instrument and server • Measurement instrument is connected to the server and is transferring data: Time the last connection was established • Measurement instrument is currently not connected to the server: Time of the last disconnection
3	The symbol indicates the signal strength. A click on the symbol displays information about the radio cell used for the last connection and the radio cells taken into account during the last positioning.

23. The measurement values are now already recorded and are immediately transmitted to Cloud server due to the "online" connection type. As soon as the first data record has been received, the time period for which the data is available is indicated in the area for displaying the available data period instead of "NO DATA".

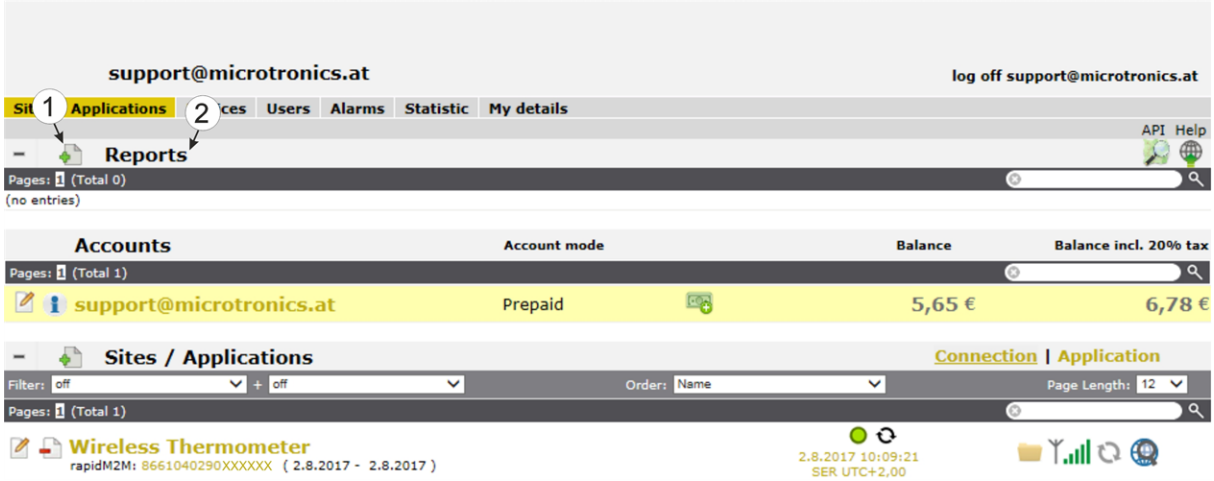


List of sites/applications

1	Display of the available data period

3.3 Phase 3: Visualising the data on the Cloud server

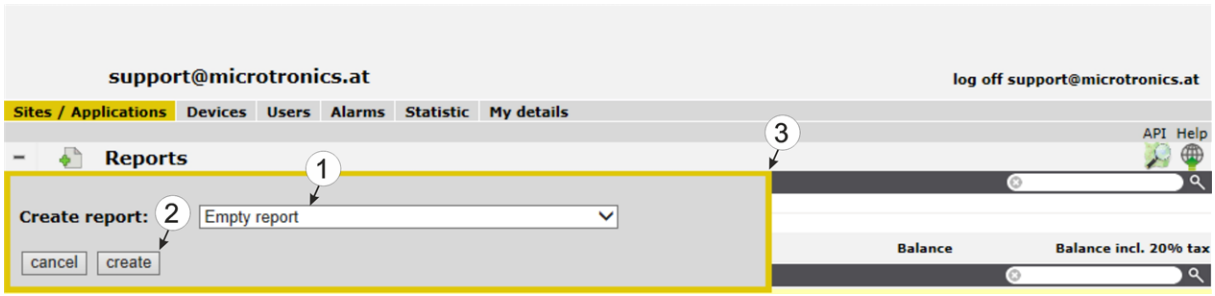
1. Create a report to display the measurement values.



"Sites/applications" area where all of the available sites are displayed.

1 Opens the input screen for creating a new report	2 List of reports

Select "Empty report" in "Create report".

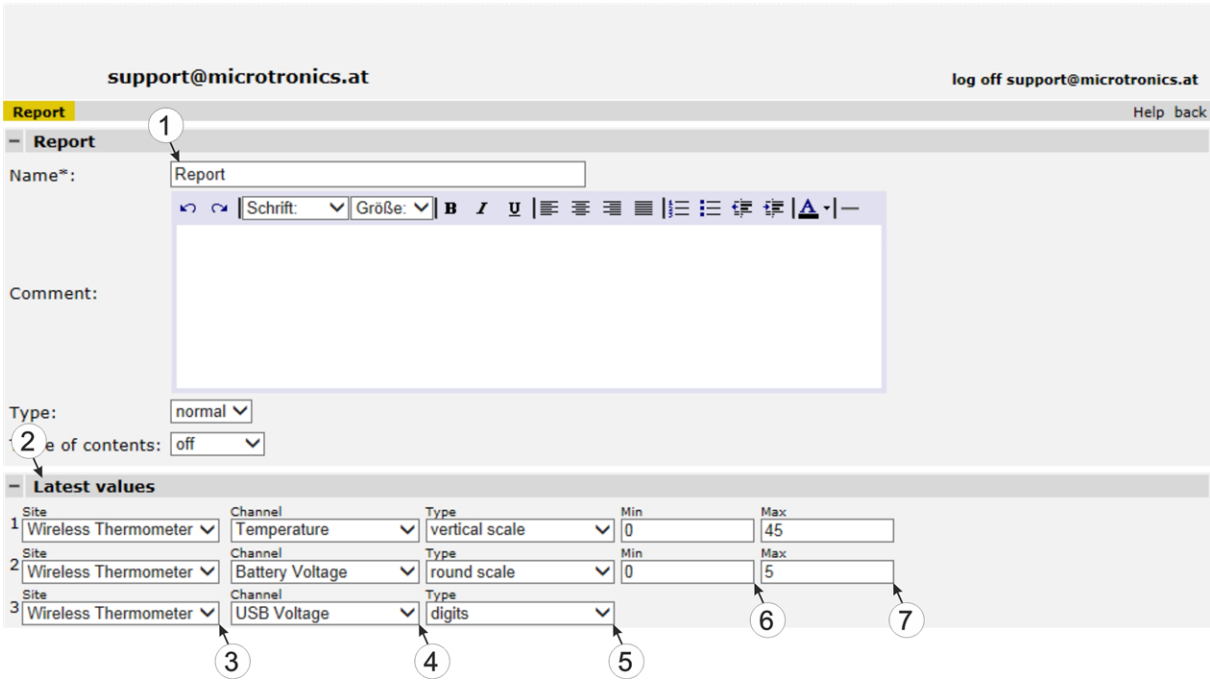


Select a report template.

1 Dropdown list of available report templates	3 Input window for selecting the report template
2 "Create" button	

2. Configure the "Last measurement values" display element as follows.. This ensures that the last measurement values of the "Temperature", "Battery voltage" and "USB voltage" measurement channels are displayed next to the name of the report in the list of reports. By clicking the "Save" button, you will return to the "Sites/applications" area where all of the available sites are displayed.

	Site	Channel	Type	Min	Max
1	Wireless thermometer	Temperature	Vertical scale	0	45
2	Wireless thermometer	Battery voltage	Round scale	0	5
3	Wireless thermometer	USB voltage	Digits		



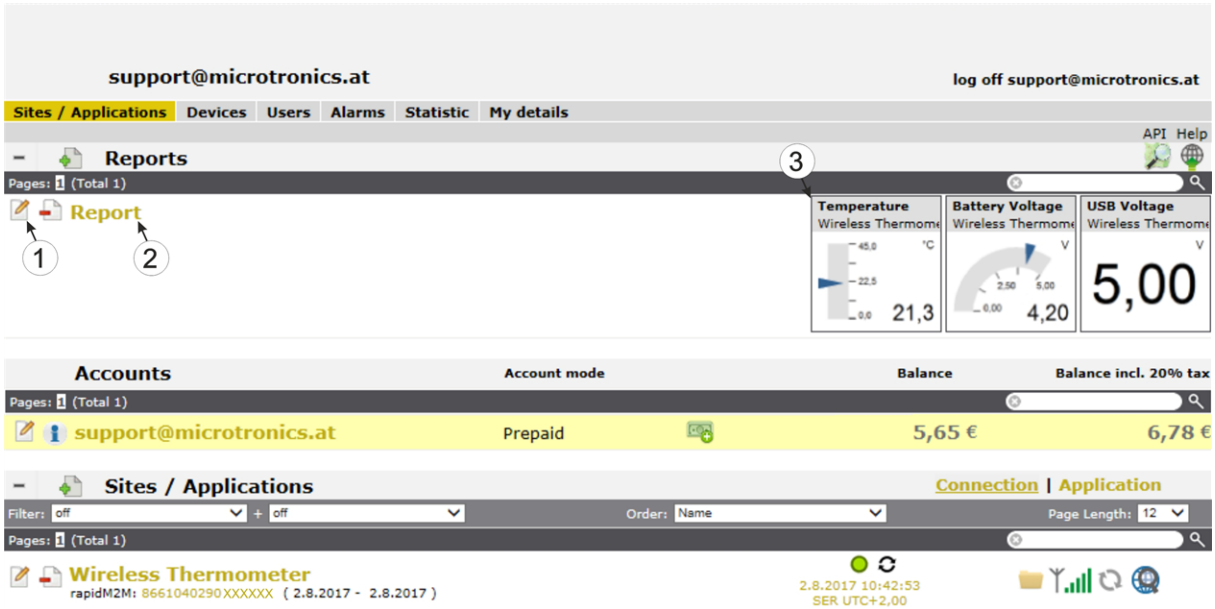
Input screen for the basic configuration of the report

1	Freely selectable name for the report
2	Configuration of the "Last measurement values" display elements that are shown next to the name of the report in the list of reports
3	Selection of the site. The element can obtain its data from different sites. All of the current customer's sites are available to be selected.
4	Site channel that should be displayed
5	Selection of the type of display element to be used
6	Lower end of the scale of the display element (only for scale and bar elements)
7	Upper end of the scale of the display element (only for scale and bar elements)

3. The current measurement value is now displayed directly next to the name of the report. The measurement values are updated according to the record interval (10 sec. if you use the value recommended in this tutorial) that you can set via the "Configuration" configuration section. The measurement values are changed by altering the temperature at the temperature sensor (e.g. using a cooling spray) or connecting/disconnecting the USB connection.

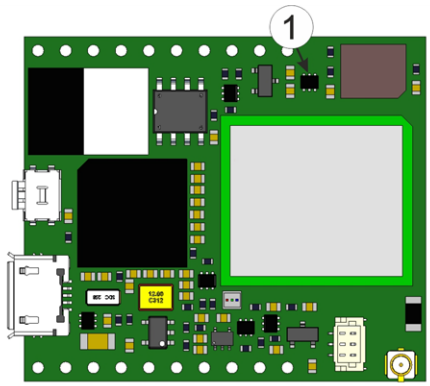
   Important note: *Please be aware that the sensitive electronic components of the the rapidM2M M3 can be damaged by static electricity if they are touched.*

   Note: *Deviations between the displayed temperature and the actual ambient temperature may occur due to the increased energy consumption associated with the "Online" connection type and subsequent self-heating of the rapidM2M M3 .*



"Sites/applications" area where all of the available sites are displayed.

1	Opens the input screen for the basic configuration of the report
2	Clicking on the name of the report will open the input screen to configure and select the elements.
3	Display elements for the last measurement value of a measurement channel (max. four per report). The site, channel and type of display element is selected in the basic configuration of the report .

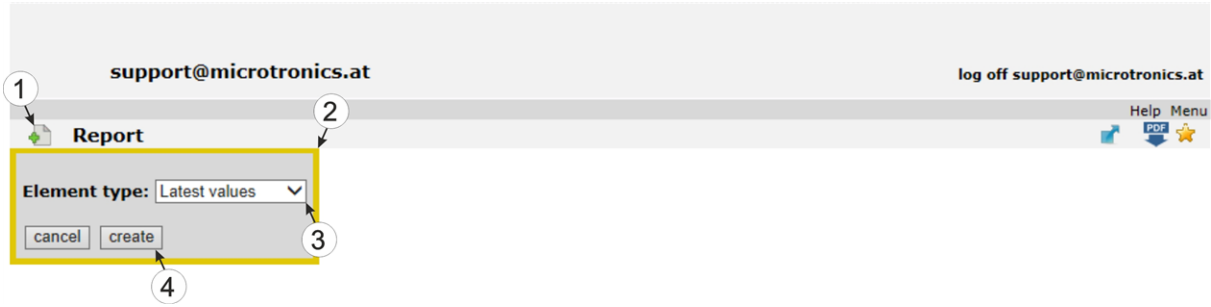


rapidM2M M3

1	Temperature sensor

4. Open the input screen to configure and select the elements by clicking on the name of the reports in the list of reports.

5. Click on the symbol to add a new element to the current report and then select "Last values" as the element type.

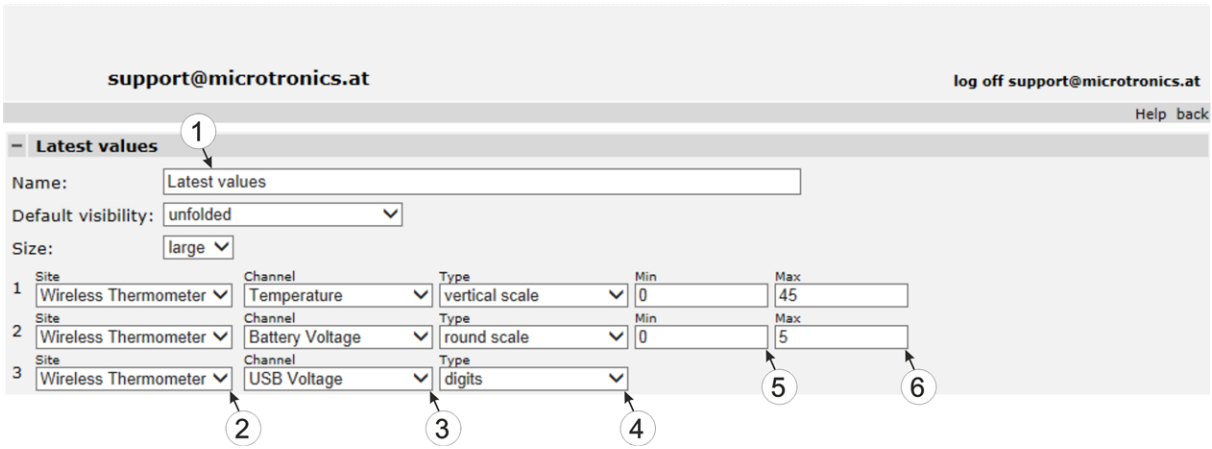


Adding a new element to the current report

1	Add a new element to the report	3	Dropdown list of available elements
2	Input window for selecting a new element	4	"Create" button

6. Configure the "Last values" element as follows. This ensures that the last measurement values for the "Temperature", "Battery voltage" and "USB voltage" measurement channels are displayed when the report is open. This is comparable to the display of the last measurement values next to the name of the report in the list of reports. Further elements, such as the graphic element described in the following step, can, however, be added to an open report. By clicking the "Save" button, you will return to the overview of elements in the report that is currently open.

	Site	Channel	Type	Min	Max
1	Wireless thermometer	Temperature	Vertical scale	0	45
2	Wireless thermometer	Battery voltage	Round scale	0	5
3	Wireless thermometer	USB voltage	Digits		



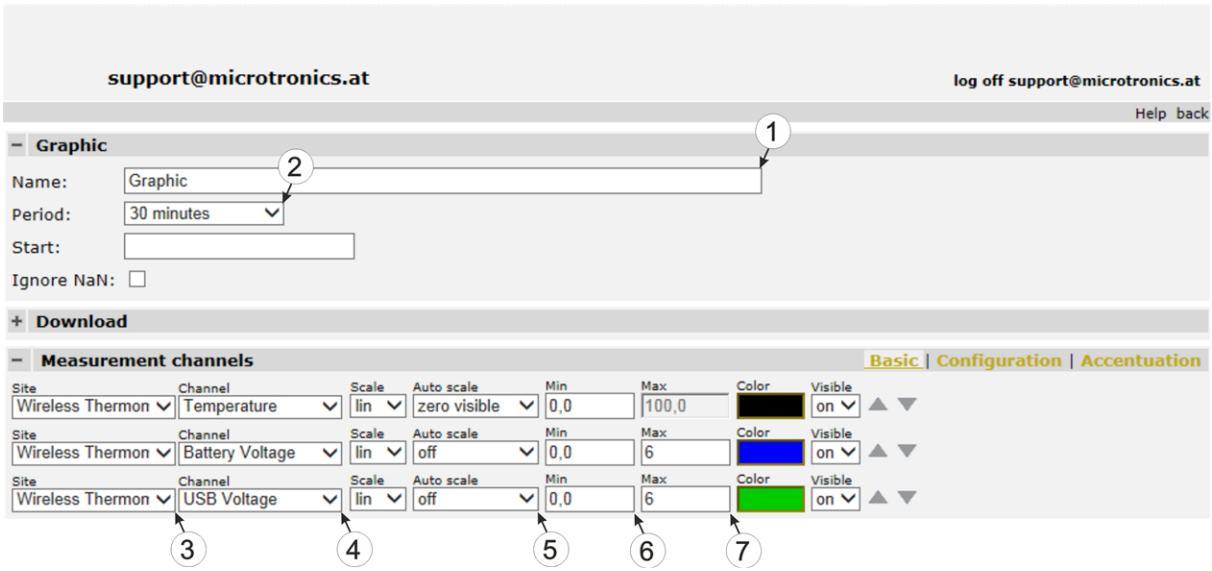
Input screen for configuring a "Last values" element type

1	Freely selectable name for the report element
2	Selection of the site. The element can obtain its data from different sites. All of the current customer's sites are available to be selected.
3	Site channel that should be displayed
4	Selection of the type of display element to be used
5	Lower end of the scale of the display element (only for scale and bar elements)
6	Upper end of the scale of the display element (only for scale and bar elements)

7. Add another element to the current report. This time, select "Graphic" as the type.

8. Configure the "Graphic" element as follows. This ensures that the progress of the measurement values for the "Temperature", "Battery voltage" and "USB voltage" measurement channels over the last 30 min. is displayed when the report is open. By clicking the "Save" button, you will return to the overview of elements in the report that is currently open.

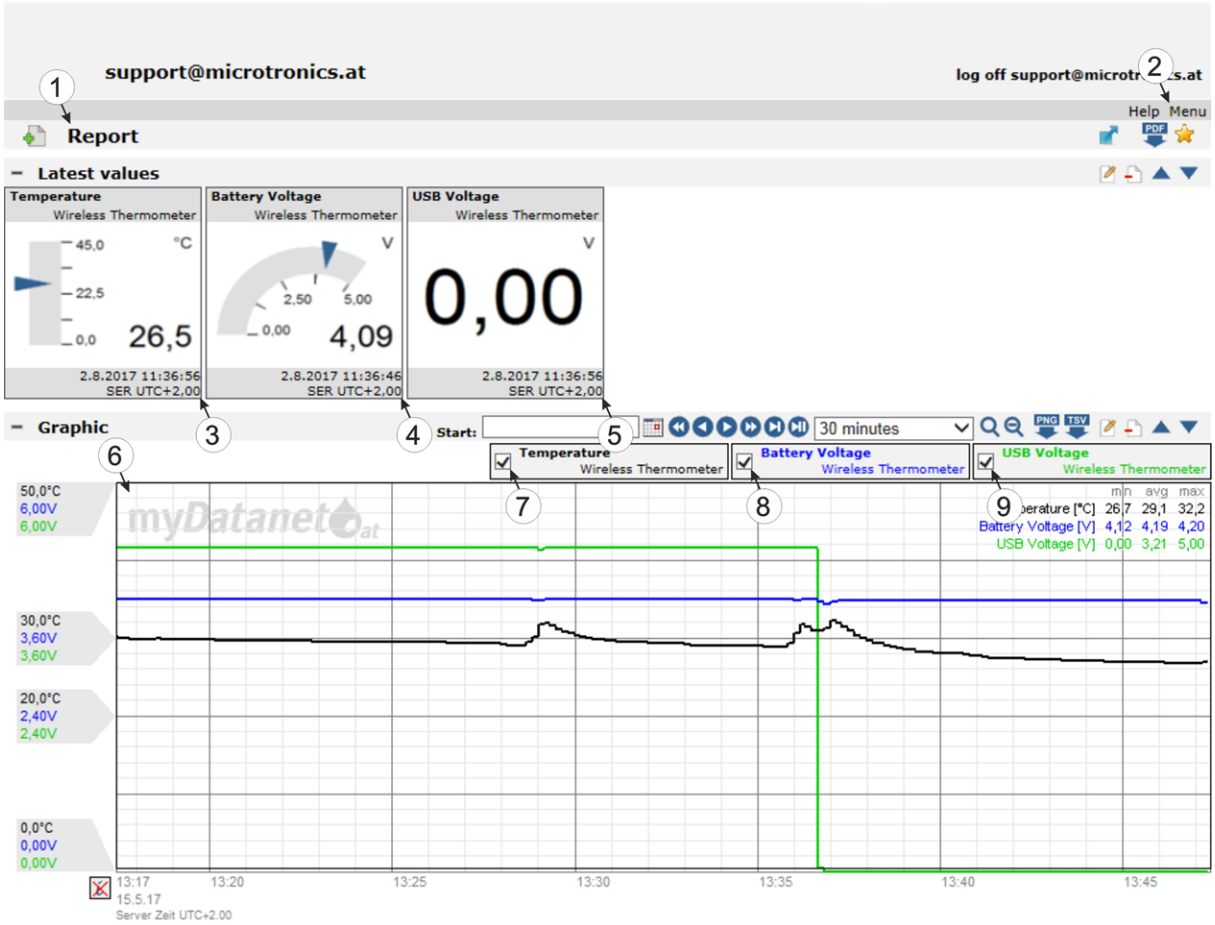
	Site	Channel	Auto scale	Min	Max
1	Wireless thermometer	Temperature	Nothing visible	0	
2	Wireless thermometer	Battery voltage	Off	0	6
3	Wireless thermometer	USB voltage	Off	0	6



Input screen for configuring a "Graphic" element type

1	Freely selectable name for the report element
2	Time period that should be displayed by default Select "30 min." here.
3	Selection of the site. The element can obtain its data from different sites. All of the current customer's sites are available to be selected.
4	Site channel that should be displayed
5	Selection of the scaling that should be used
6	When "Auto Scale" "off": Lower end of the scale When "Auto Scale" is "Floating" or "Zero visible": Minimum scale size (difference between the ends of the scale)
7	Upper end of the scale

9. Due to the "online" connection type, the determined measurement values are immediately transmitted to the Cloud server and can be visualised using the report created in the previous steps. The display of the "Last values" element is automatically updated upon receipt of new data. The measurement value graphic of the "Graphic" element, on the other hand, requires user interaction (e.g. clicking on or moving the measurement value graphic).



Visualising the measurement data via the report

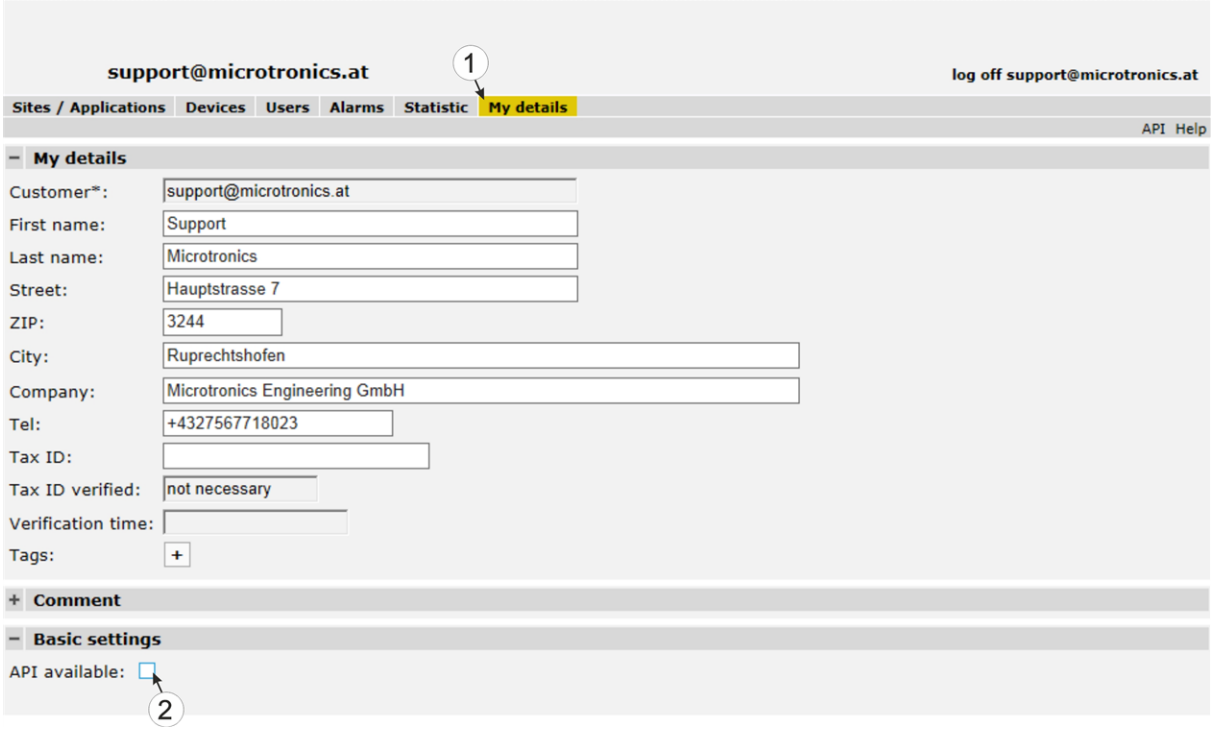
1	Name of the report used to visualise the measurement data
2	Closes the report and returns to the "Sites/applications" area
3	Last measurement value of the "Temperature" measurement channel
4	Last measurement value of the "Battery voltage" measurement channel
5	Last measurement value of the "USB voltage" measurement channel
6	Measurement value graphic (displays the measurement values over time)
7	Checkbox to display and hide the "Temperature" measurement channel
8	Checkbox to display and hide the "Battery voltage" measurement channel
9	Checkbox to display and hide the "USB voltage" measurement channel

3.4 Phase 4: Access to the data and configurations via the REST-API of the Cloud server using the rapidM2M Playground

> ***Important note:*** *The API is a chargeable feature. Monthly costs are thus generated per site/application. The amount is always calculated on the last day of the month and subtracted from the balance.*

This phase is optional and is designed to help familiarise yourself with the resources provided by the API of the Cloud server and how they are handled.

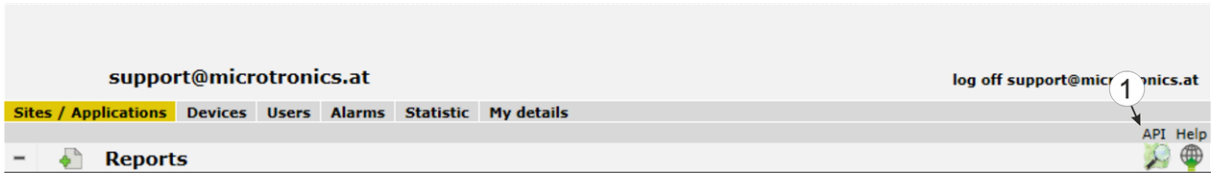
1. Switch to the "My details" area and activate the "API available" checkbox in the "Basic settings" configuration section. When clicking on the "Save" button, you will once again be reminded that the API is a chargeable feature.



Overview of the "My details" area

1	Button to switch to the "My details" area
2	Checkbox to activate the API

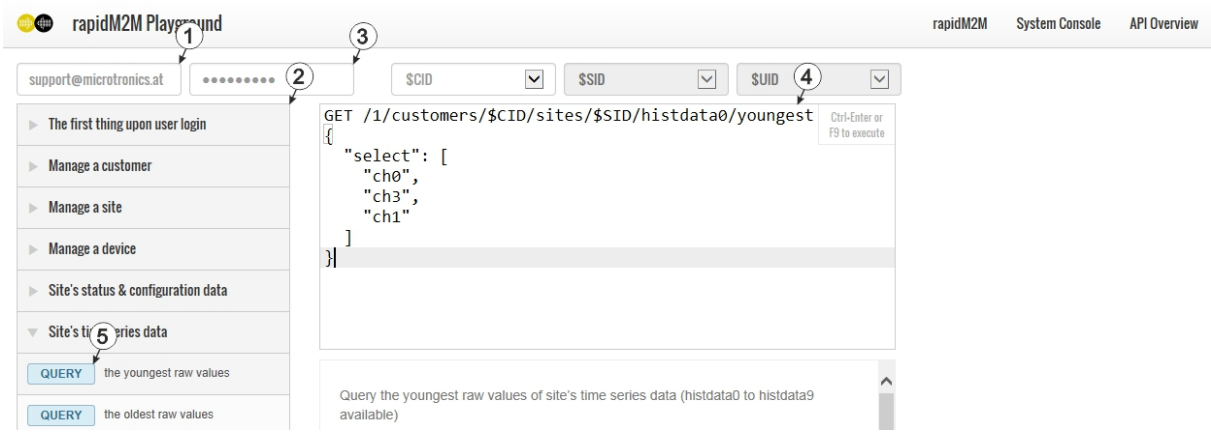
2. Switch back to the "Sites/applications" area and click on "API" to open the rapidM2M Playground . The rapidM2M Playground enables you to familiarise yourself with the API of the Cloud server and to test the provided functions.



"Sites/applications" area where all of the available sites are displayed.

1	Opens the rapidM2M Playground

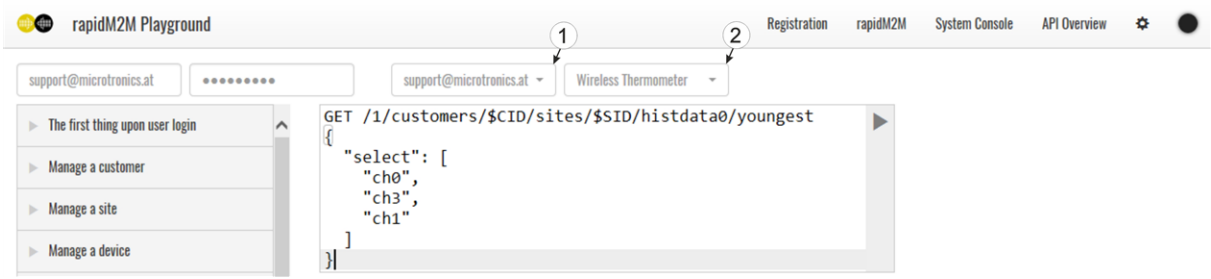
3. Enter the login credentials and select the "QUERY the youngest raw values" HTTP command from the "Site's time series data" group. This command returns the latest saved measurement data record. You can use the same login credentials for the rapidM2M Playground as you use for the Cloud server.



Selecting the HTTP command"QUERY the youngest raw values"

1	Input field for the user name
2	List of the available HTTP commands. The HTTP commands are grouped according to their fields of application.
3	Input field for the password
4	Window displaying the selected HTTP command
5	HTTP command to query the latest saved measurement data record.

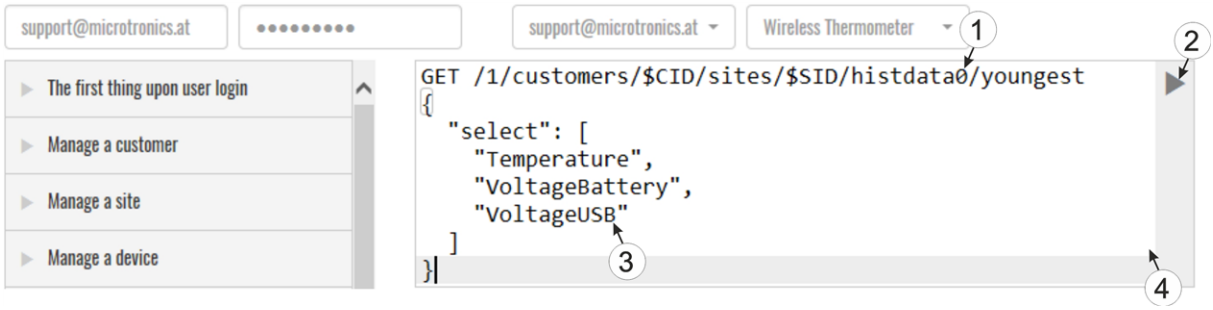
4. Some of the resource paths of HTTP command contain wildcards (e.g. $CID for the customer). The currently selected "QUERY the youngest raw values" HTTP command requires a customer and site to be specified. During the registration, a customer is always created on the Cloud server at the same time. This customer is automatically assigned the same name as the user. The serial number of the device is automatically used as the name for the site during the creation. Although the name of the site was changed to "Wireless thermometer" during the last step of phase 1.



Selection of the customer and site that should replace the wildcards in the resource path

1	Selection of the customer that shall replace the "$CID" wildcard in the resource path of the HTTP command Select your user name here. (A customer with the same name as your user was created during registration)
2	Selection of the site that shall replace the "$SID" wildcard in the resource path of the HTTP command Select "Wireless thermometer" here.

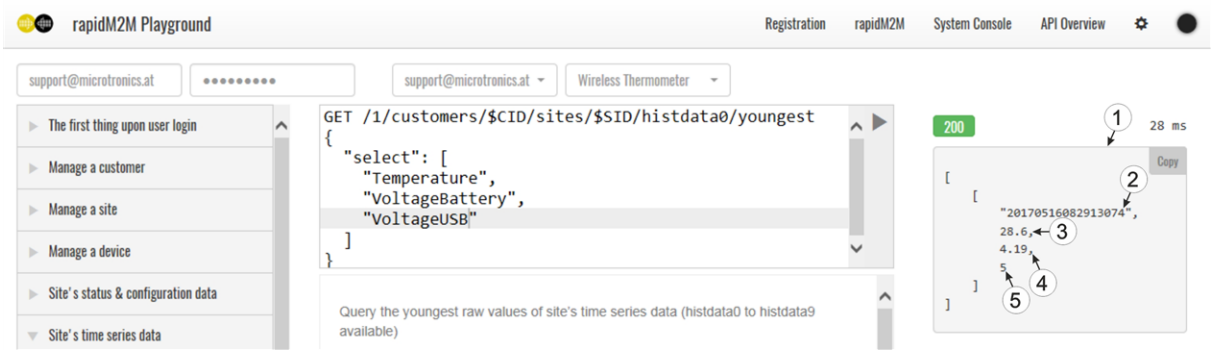
5. To be able to retrieve the measurement data, the HTTP command must be modified as illustrated in the following figure. The alternative field name (alias) specified in step 10 of phase 1 must be used for every measurement value that should be read. Note that no comma should be added after the last field name. Then click on the button to execute the HTTP command.



Modifying the HTTP command "QUERY the youngest raw values"

1	Measurement data channel of the Cloud server that should be accessed The use of measurement data channel 0 (histdata0) was specified in step 9 of phase 1.
2	Button to execute the HTTP command
3	List of measurement values of measurement data channel 0 for which the values should be retrieved The alternative field names (alias) that should be used here were specified in step 10 of phase 1.
4	Window displaying the selected HTTP command

6. The answer to the HTTP command is displayed in the right window. The "QUERY the youngest raw values" HTTP command reads out the last measurement data record saved in the database of the Cloud server. The values thus correspond exactly to those issued by the display elements of the report created in phase 3.



Result of the HTTP command"QUERY the youngest raw values"

1	Window displaying the JSON object that is generated as a response to the HTTP command
2	Time stamp of the measurement data record
3	Last measurement value of the "Temperature" measurement channel
4	Last measurement value of the "Battery voltage" measurement channel
5	Last measurement value of the "USB voltage" measurement channel

7. The "QUERY a site's configuration" HTTP command is used to query the configurations. The HTTP command does not need to be modified as the decision was made to use configuration block 0 (config0) during step 8 of phase 1. After selecting the HTTP command, click directly on the button to execute the HTTP command.

Selecting the HTTP command"QUERY a site's configuration"

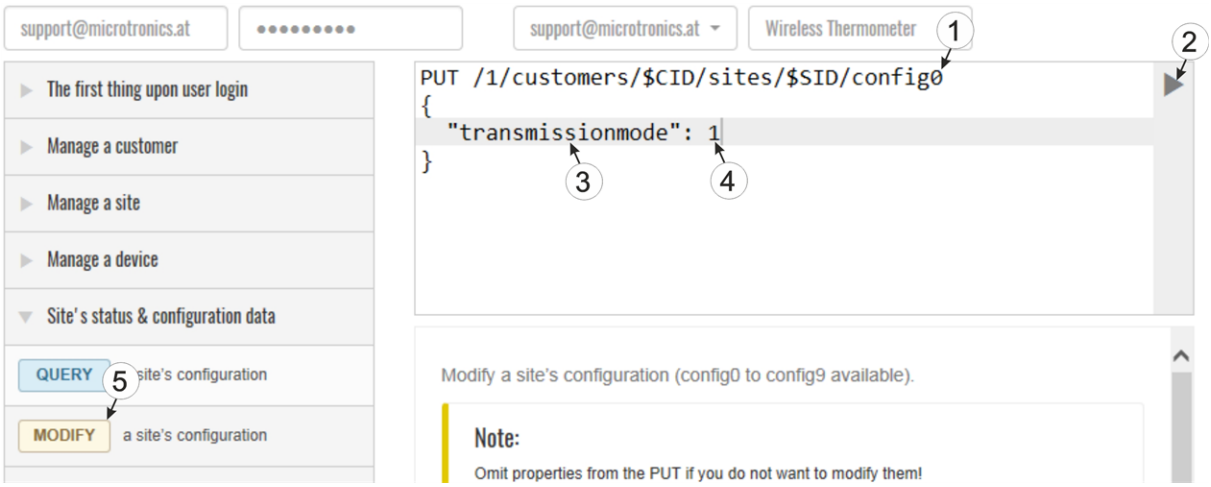
1	Configuration block that should be accessed The use of configuration block 0 (config0) was specified in step 8 of phase 1.
2	Button to execute the HTTP command
3	HTTP command to query one of the configuration blocks
4	Window displaying the selected HTTP command

8. The answer to the HTTP command is displayed in the right window.

Result of the HTTP command"QUERY a site's configuration"

1	Window displaying the JSON object that is generated as a response to the HTTP command
2	Time stamp of the configuration block
3	Time between measurement data recordings
4	Time between transmissions (only applies to the "Interval" and "Wakeup" connection types)
5	Selection of the connection type (0 = Interval, 1 = Wakeup, 2 = Online)

9. The values of the configuration parameters can be adjusted via the "MODIFY a site's configuration" HTTP command. In this specific example, the connection type should be changed from "Online" to "Wakeup". The alternative field name (alias) must be known to be able to modify a parameter. It is possible to change both individual and multiple configuration parameters by calling up the "MODIFY a site's configuration" HTTP command. Note that no comma should be added after the last field name. Following the selection, modify the HTTP command as illustrated in the following figure and then click on the button to execute the HTTP command.



Modifying the HTTP command "MODIFY a site's configuration"

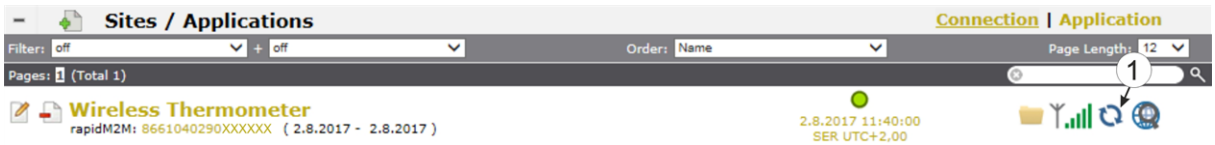
1	Configuration block that should be accessed The use of configuration block 0 (config0) was specified in step 8 of phase 1.
2	Button to execute the HTTP command
3	Alternative field name (alias) of the parameter in configuration block 0 that should be modified The alternative field name (alias) that should be used here was specified in step 8 of phase 1.
4	Value that should be set Select "1" here for the "Wakeup" connection type. The device connects in the transmission cycle.However, a connection can also be initiated through the server. 
5	HTTP command to modify one of the configuration blocks

10. The window in which the JSON object is displayed is empty as the "MODIFY a site's configuration" HTTP command does not provide a response body. The successful execution of the HTTP command is indicated by response code 204 in the right-hand area.

Result of the HTTP command"MODIFY a site's configuration"

1	Response code sent by the Cloud server as an answer to the HTTP command
2	Window displaying the JSON object that is generated as a response to the HTTP command

11. Due to the "online" connection type, any changes to the configuration implemented by the "MODIFY a site's configuration" HTTP command are immediately transmitted to the rapidM2M M3 . In this particular case, the change causes the device to switch to the "Wakeup" connection mode. On the one hand, this is indicated by the LED going out and, on the other, that the rotating arrows are hidden in the list of sites/applications. The symbol to send a wakeup SMS is also displayed in the list of sites/applications. Upon receipt of such an SMS, the device will immediately establish a connection to the server.



List of sites/applications

1	Sends a wakeup SMS to the rapidM2M M3 to instruct it to establish an immediate connection to the Cloud ***Important note:*** *Costs are incurred by sending a wakeup SMS.*

12. To continue this tutorial, the rapidM2M M3 must be switched back to the "Online" connection mode. Use the "MODIFY a site's configuration" HTTP command again to do this. Following the selection, modify the HTTP command as illustrated in the following figure and then click on the button to execute the HTTP command.

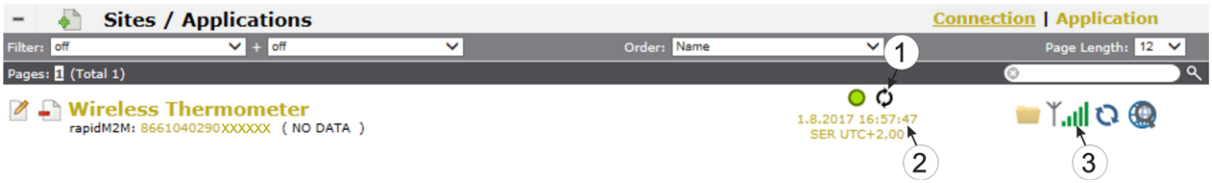
Modifying the HTTP command "MODIFY a site's configuration"

1	Configuration block that should be accessed
2	Button to execute the HTTP command
3	Alternative field name (alias) of the parameter in configuration block o that should be modified
4	Value that should be set Select "1" here for the "Online" connection type. The device does not disconnect the connection and continuously transmits the measurement data.
5	HTTP command to modify one of the configuration blocks

13. The successful execution of the HTTP command is once again indicated by response code 204 in the right-hand area.

14. There is no active communication between the Cloud server and the rapidM2M M3 during the "Wakeup" connection mode. The configuration changes implemented via the "MODIFY a site's configuration" HTTP command have thus not yet been transmitted to the device. Therefore, press the button of the rapidM2M M3 to establish a connection and subsequently synchronise the configurations.

15. Switch to the Cloud server and wait until the list of sites/applications indicates that the rapidM2M M3 is connected to the Cloud server (rotating arrows). This can take up to two minutes.

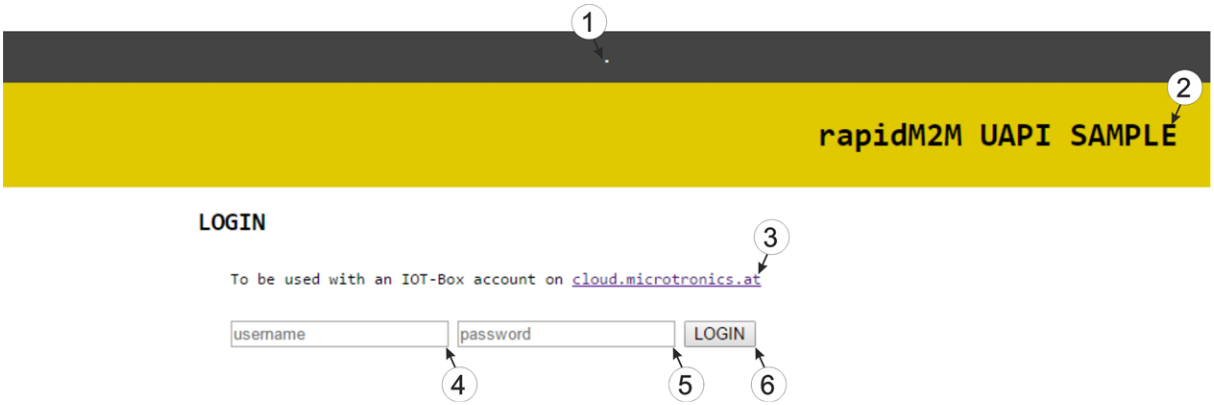


List of sites/applications

1	Indicates that the rapidM2M M3 is connected to the server and is transmitting data. This symbol is hidden when the connection is disconnected.
2	Information regarding the time of the communication between the measurement instrument and server • Measurement instrument is connected to the server and is transferring data: Time the last connection was established • Measurement instrument is currently not connected to the server: Time of the last disconnection
3	The symbol indicates the signal strength. A click on the symbol displays information about the radio cell used for the last connection and the radio cells taken into account during the last positioning.

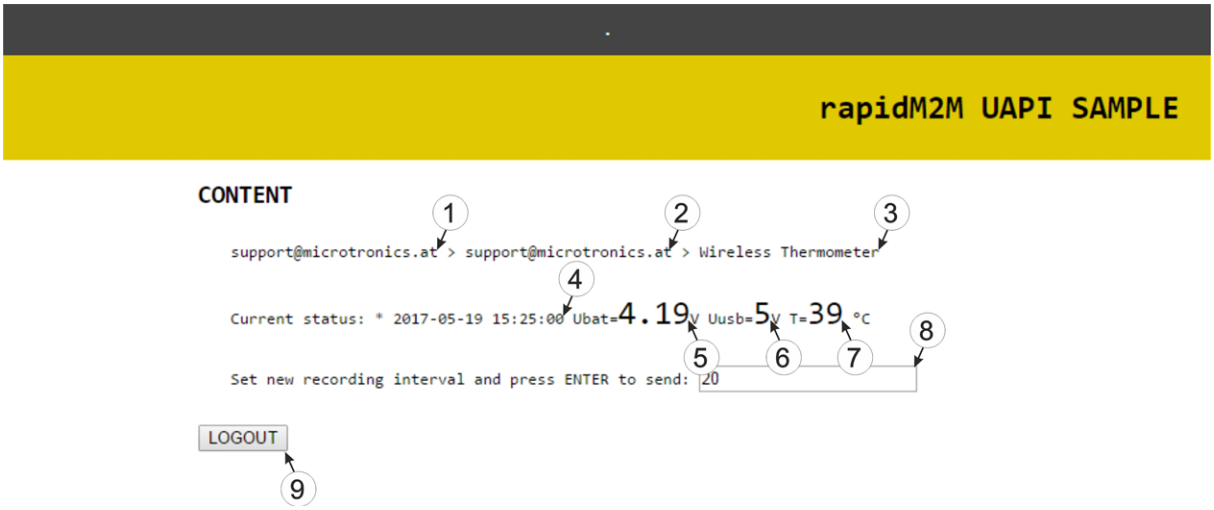
3.5 Phase 5: Creating your own dashboard

Note: *Some of the functions used are not supported by all browsers. The use of "Google Chrome" from version 58 onwards is thus recommended.*



Dashboard login page

1	Status indication	4	Input field for the user name (e-mail address)
2	Name of the dashboard	5	Input field for the password
3	Web address of the Cloud server	6	Checks the user credentials and switches to the content page

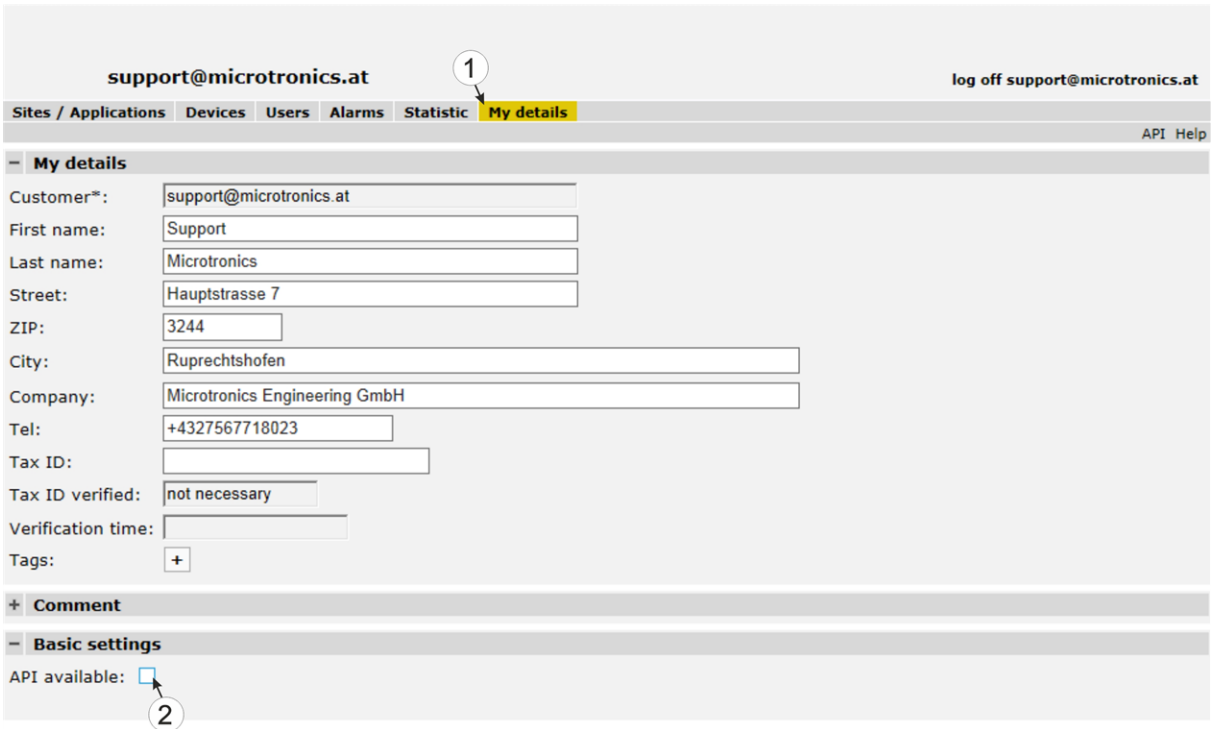


Dashboard content page

1	Name of the customer (e-mail address specified when registering the account)	6	Last measurement value of the "USB voltage" measurement channel
2	User name (e-mail address)	7	Last measurement value of the "Temperature" measurement channel
3	Name of the site	8	Input field for the time between the measurement data recordings (adopted when pressing the "Enter" button)
4	Time stamp of the measurement data record	9	Logs out the active user and switches to the login page
5	Last measurement value of the "Battery voltage" measurement channel		

The first step is only necessary if you have skipped phase 4 of this tutorial.

1. Switch to the "My details" area and activate the "API available" checkbox in the "Basic settings" configuration section. When clicking on the "Save" button, you will once again be reminded that the API is a chargeable feature.



Overview of the "My details" area

1	Button to switch to the "My details" area
2	Checkbox to activate the API

2. Create a new empty html file with UTF-8 coding.

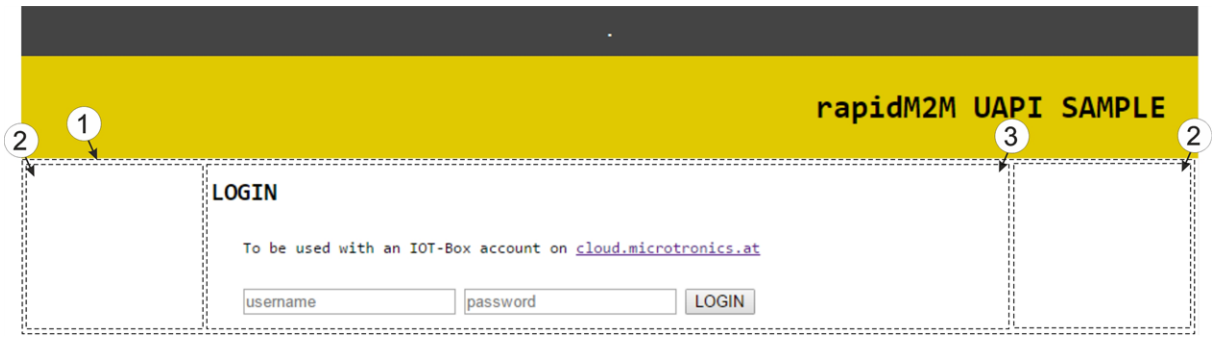
> ***Note:*** *If you would like to skip the step-by-step creation of the dashboard, open the "Dashboard.html" file with a browser instead of creating a new html file and continue the tutorial at step 11 . The "Dashboard.html" file is included in the package of additional files for this tutorial.*

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <title> </title>
  </head>

  <body>

  </body>
</html>
```

3. You must first create the header of the html file. In addition to the title of the file, it contains the formatting instructions for the tags and classes.

```
<head>
  <!-- Set character encoding to UTF-8 -->
  <meta charset="utf-8">
  <!-- Title of the file, which, amongst other things, is displayed on the title bar of the browser window
                                                                                                        -->
  <title>rapidM2M UAPI SAMPLE for user agents</title>
  <style>
    /* Body: use full width of browser window, "Monospace" font, no margins */
    body   { width:100%;font-family:Monospace;margin:0}
    /* Dashboard name: no margin, yellow background, right-aligned, internal spacing = 1x font size */
    h1     { margin:0;background:#e0c900;text-align:right;padding:1em}
    /* Master: activates the use of the flex properties for the sub-elements of this area */
    .master{ display:flex }
    /* Spacer: The area takes over the remaining available room of an area by a factor of 1 */
    .spacer{ flex-grow:1 }
    /* Login page or content page: hidden during initialisation, width = 50x font size  */
    .page  { display: none; width:50em}
    /*All of the DIVs within the login page or content page: Margin = 2x font size */
    .page>div{ margin:2em }
    /*  Status display: white font on black background, centred, internal spacing = 1x font size */
    #uapi_msg{ background:#444;color:white;text-align:center;padding:1em}
    /*  Measurement value display: Font size = 2x content page standard font size, standard font */
    em { font-size:2em;font-style:normal; }
  </style>
</head>
```

> ***Note:*** *A "master" class area is initially created to display the content of the login page or content page in the centre of the browser window. The use of the flex properties is activated for its sub-elements. One area of the "spacer" class, two areas of the "page" class (for the login page and content page, whereby only one of the two is always displayed) followed by another area of the "spacer" class are created successively as a sub-element of the "master" area. A fixed width has been defined for the "page" class. The "flex-grow:1" property for the "spacer" class stipulates that any remaining, available room within the "master" area is distributed evenly between the two "spacer" class areas. This ensures that the content of the login page and content page is always automatically aligned centrally even if the size of the browser window changes.*



Additional explanation regarding the structure of the dashboard display area

1	"master" class area	3	"page" class area
2	"spacer" class area		

4. The dashboard user interface must be created next. It is located in the body section of the html file along with the JavaScript code. The creation of the JavaScript code has been split across the next five steps to provide a better overview. The code must be added below the comment line "*<!-- JavaScript code (see step 5 to 9) -->*".

> ***Note:*** *Please note that the login page and content page are initially hidden due to the definition of the "page" class and are only displayed by the JavaScript code depending on the situation.*

```
<body>
  <div id="uapi_msg">.</div>           <!-- Status indication -->

  <h1>rapidM2M UAPI SAMPLE</h1>        <!-- Name of the dashboard -->
  <div class="master">                 <!-- for centre alignment if the size of the window changes -->
    <div class="spacer"></div>         <!-- Spacer on left side of the login page or content page -->
    <div id="pg_login" class="page">   <!-- Login page -->
      <h2>LOGIN</h2>                   <!-- Title of the login page -->
      <div>                            <!-- Display of the server web address -->
        To be used with an IOT-Box account on
        <a href="https://cloud.microtronics.at">cloud.microtronics.at</a>
      </div>
      <div>
        <!-- Input field for the user name, set placeholder to "username", field is focused -->

        <input id="login_usr" type="text" placeholder="username" autofocus></input>
        <!-- Input field for the password, set placeholder to "password" -->
        <input id="login_pwd" type="password" placeholder="password"></input>
        <!-- Login button to check the user credentials and switch to the content page -->
        <button id="btn_login">LOGIN</button>
      </div>
    </div>

    <div id="pg_content" class="page"> <!-- Content page -->
      <h2>CONTENT</h2>                 <!-- Title of the content page -->
      <div id="me">loading...</div>    <!-- Display of customer, user and site name -->
      <div>                            <!-- Display of the last measurement values (incl. time stamp) -->
         Current status: <span id="status">loading...</span>
      </div>
      <div>
        <!-- Input field for the record interval, placeholder "new value + ENTER", field is focused -->
        Set new recording interval and press ENTER to send:
        <input id="ed_reciv" type="text" placeholder="new value + ENTER" autofocus></input>
      </div>

      <!-- Button to log out the active user and switch to the login page -->
      <button id="btn_logout">LOGOUT</button>
    </div>
    <div class="spacer"></div>         <!-- Spacer on the right side of the login page or content page -->
  </div>
  <!-- Include the jQuery library  -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <!-- Include the rapidM2M UAPI library -->
  <script src="https://cdn.microtronics.at/libs/fe/rapidm2m-uapi.js"></script>
  <script>

    <!-- JavaScript code (see step 5 to 9) -->

  </script>
</body>
```

5. A few constants and global variables and two simple functions for the program sequence must be defined next.

```
"use strict";                       // The JavaScript code should be executed in "strict mode".

showpage( '#pg_login');             // Display login page

// Constant that contains the server web address
const MY_HOST = 'https://cloud.microtronics.at/api';

/*=============================================================================================
Function to hide/display the login page and content page

pg:[in]
  ID of the page that should be displayed
=============================================================================================*/
function showpage(pg){
  $('.page').hide();                // Hide all pages
  $(pg).show();                     // Display page for which the ID was transmitted
  // Searches the page for an object with a set "autofocus" attribute and sets the focus on this object
  $(pg).find('[autofocus]').focus();
}

let uapi;         // Variable for a new instance of the JavaScript class "UAPI"
let poll_tmr;     // Variable to store the ID returned by the "setTimeout" function
let poll_spin=0;  // Variable to toggle the symbol to display the activity in the content page

// Constant for the js object that refers to the DOM element of the "Record interval" input field
const $ed_reciv= $('#ed_reciv');

/*=============================================================================================
Shows the transferred string in the status indication area. The characters "." is displayed
in the status indication if an empty string or no string is transmitted.

msg:[in]
  String that should be displayed in the status indication area
=============================================================================================*/
function msg(msg){
  $('#uapi_msg').html( msg || '.');
}
```

6. The function that updates the display of the last measurement values at one second intervals must then be created.

```
/*=============================================================================================
Updates the area in which the last measurement values are displayed
Once the API query is issued, this function sets a timeout of 1 sec. following which
it is called up again. This function is called up for the first time
once the user credentials have been checked successfully when clicking on the login button
=============================================================================================*/
function poll_status(){

  /* Constant for the list of measurement values (alias), for which the current value should be read out
     from the server                                                                              */
  const o = {
              select:["VoltageBattery","VoltageUSB","Temperature"]
            };
  /* Get access to the latest saved measurement data records of measurement data channel 0, whereby the
     measurement values specified in constant "o" are read out. The anonymous function processes the
     response code (err) and the returned JSON array(x)                                           */
  uapi.get( '1/customers/$cid/sites/$sid/histdata0/youngest', o, (err,x)=>{

    /* Variable to store the string displayed in the area for displaying the last measurement values.
       The first character for each call-up alternates between "*" and "." to indicate that
       there is activity                                                                          */
    let s = (++poll_spin & 1) ? '* ':'. ';

    if (err)                        // If an error has occurred ->
      s += err                      // Add error code to string "s"
    else if (!x.length)             // Otherwise -> If the JSON array has a length of 0 ->
      s += '(no data yet available)';//Add note to string "s" indicating that no data is available
    else {
      /* Convert time stamp from the JSON array element [0][0] to the "yyyy-mm-dd hh:nn:ss[.zzz]" format
         , delete the "[.zzz]" part and add the result to string "s"                              */
      s += uapi.stampFormat( x[0][0]).substr(0,19);
      // Add the measurement values from the JSON array incl. corresponding inscription to string "s"
      s += ` Ubat=<em>${x[0][1]}</em>V Uusb=<em>${x[0][2]}</em>V T=<em>${x[0][3]}</em> &deg;C`;
    }

    $('#status').html( s);          // Display string "s" in the area for the last measurement values

    /* Set timeout of 1000 ms and save returned ID in the variable. The "poll_status" function
       is called up again following expiry of the timeout.                                        */
    poll_tmr= setTimeout( poll_status, 1000);
  });

}
```

7. You must now specify which actions are executed when clicking on the login button.

```
/*=============================================================================================
When clicking on the login button, the user credentials are initially checked and the
first site within the first customer, for whom the user has access rights,
is selected. The current record interval is then read out from the server,
the content page is displayed and the function to update the area with
the last measurement values is called up for the first time. This function then calls
itself up repeatedly by setting a timeout at one second intervals.
=============================================================================================*/
$('#btn_login').click(()=>{

  msg('connecting...');            // Display the text "connecting..." in the status indication area

  uapi=new UAPI(                   // Create a new instance of the "UAPI" JavaScript class and
    $('#login_usr').val(),         // copy the user name from the input field into the class
    $('#login_pwd').val(),         // copy the password from the input field into the class
    MY_HOST);                      // Copy the web address of the server from the constant into the class

  // Determine global error signalling
  uapi.onError= (err,info)=>{      // Adopt callback into the instance of the "UAPI" JavaScript class
    switch(err) {                  // Switch the response code (err) ->
      // Invalid user credentials -> Display "Illegal user credentials!" in the status indication area
      case 401: msg('Illegal user credentials!'); break;
      // For all other errors -> Display response code and additional information in the status indication
      default : msg('ERR#'+err +' ... ' + info);
    }
  }

  /* Determine list of customers, for which the current user has access rights. The anonymous function
     processes the response code (err) and the returned JSON object (me) including the customer list    */
  uapi.get('1/me',(err,me)=>{

    if (err) return;               // In the event of an error, remain on the login page
                                   // (everything else is dealt with by global error signalling.)

    /* Select first customer from the list. In the following resource paths, "$cid" is replaced by
       the customer name adopted in the "uapi" instance.                                            */
    uapi.placeholders.$cid= me.customers[0];

    /* Determine list of sites assigned to the selected customer. The anonymous function
     processes the response code (err) and the JSON object (sites) including the list of sites          */
    uapi.get('1/customers/$cid/sites',(err,sites)=>{

      if (err) return;             // In the event of an error, remain on the login page
                                   // (everything else is dealt with by global error signalling.)

      /* Select first site from the list. In the following resource paths, "$sid" is replaced by
       the unique site ID adopted in the "uapi" instance.                                           */
      uapi.placeholders.$sid= sites[0]._uid;

      /* Read out config 0 of the selected site. The anonymous function processes the response code
         (err) and the JSON object(x) including the configuration parameters.                       */
      uapi.get( '1/customers/$cid/sites/$sid/config0',(err,x)=>{

        if (err) return;           // In the event of an error, remain on the login page
                                   // (everything else is dealt with by global error signalling.)

        // Write the record interval read out from the server in the input field for the record interval
        $ed_reciv.val( x.recordinterval);

        msg();                    //Delete the area for the status indication

        showpage('#pg_content');  //Display content page and simultaneously hide the login page

        //Display the currently selected customer, user and site names in the corresponding area
        $('#me').text(
          me.user.email + ' > ' +
          me.customers[0] + ' > ' +
          sites[0].name);

        //Call up the function to update the area with the last measurement values for the first time
        poll_status();
      });
    });
  });
});
```

8. It must then be specified, how the input of a changed record interval is handled.

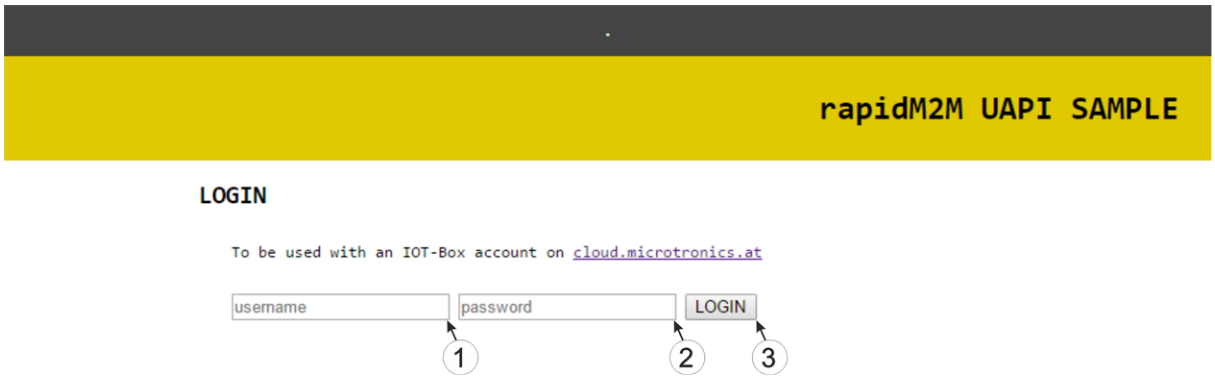
```
/*=============================================================================================
If the cursor is in the input field for the record interval and the ENTER button
is pressed, the value entered in the input field is copied to config 0 and
transmitted to the server. This also ensures that the minimum interval
is observed.
=============================================================================================*/
//When a key is released, the anonymous function processes the event object
$ed_reciv.on('keyup', (ev)=>{

  if (ev.keyCode !== 13) return;  // If it is not the ENTER key -> return

  msg('Saving...');               // Display the text "Saving..." in the status indication area

  //Constant that specifies the parameter (alias) and the value to be set
  const o= {
             //If there are invalid entries or values at < 10 sec., set the record interval to 10 sec
             record interval: Math.max( $ed_reciv.val() || 10, 10)
           }

  /* Use PUT to update config 0 for the site selected during login, whereby constant "o"
     specifies the parameter and the value to be set. The anonymous function processes the
     response code (err)                                                                    */
  uapi.put( '1/customers/$cid/sites/$sid/config0', o, (err)=>{

    if (err) return;               // In the event of an error -> return
                                   // (everything else is dealt with by global error signalling.)

    msg();                         //Delete status indication area
  });
});
```

9. Finally, it is necessary to define how the dashboard should react when clicking on the logout button.

```
/*=============================================================================================
When clicking on the logout button, the second-by-second update of the area with
the last measurement values is stopped, the instance of the "UAPI" JavaScript class,
that contains the latest user credentials, is deleted and the login page is displayed.
=============================================================================================*/
$('#btn_logout').click(()=>{

  /* Deleting the timeout interrupts the loop during which the function continuously re-initiates itself
     to update the area with the last measurement values.                                   */
  clearTimeout( poll_tmr);

  uapi= null;                      // Delete the instance of the "UAPI" class including the user
                                   // credentials
  showpage('#pg_login');           // Display the login page and simultaneously hide the content page
});
```

10. Save the created file and then open it using a browser.

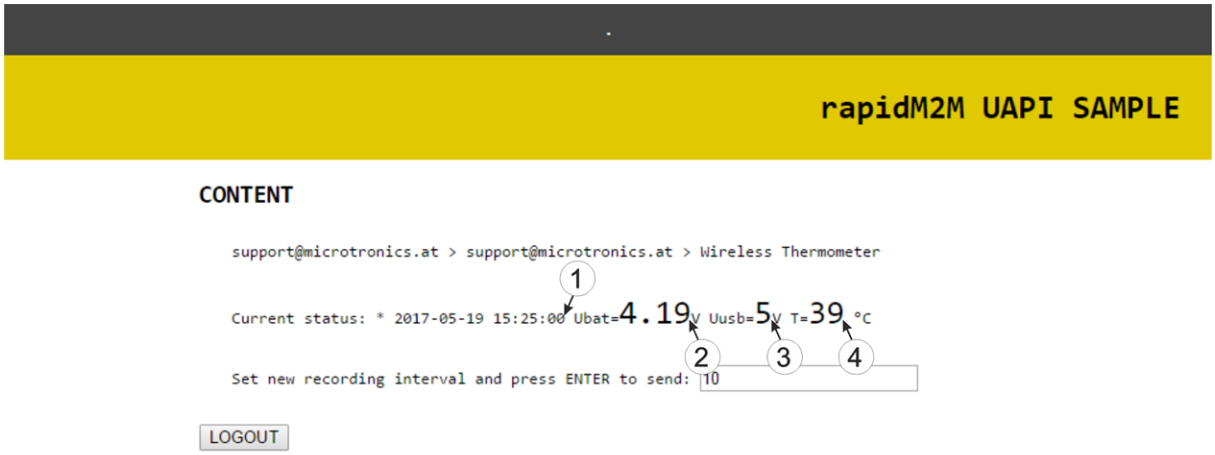
11. Enter your user name (e-mail address) and your password in the fields provided and click on the login button.



Dashboard login page

1	Input field for the user name (e-mail address)	3	Checks the user credentials and switches to the content page
2	Input field for the password		

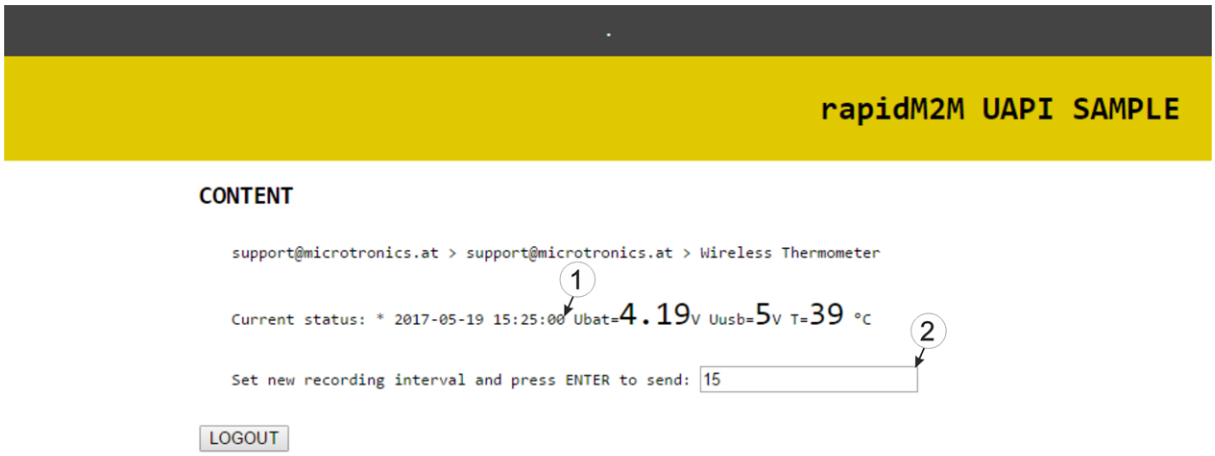
12. The content page that is now open shows the last determined measurement values and the time when the measurement values were generated. If you use the value of 10 sec. recommended in this tutorial, the time of the measurement always changes by 10 sec.



Dashboard content page

1	Time stamp of the measurement data record	3	Last measurement value of the "USB voltage" measurement channel
2	Last measurement value of the "Battery voltage" measurement channel	4	Last measurement value of the "Temperature" measurement channel

13. Change the record interval to 15 and exit the input field by pressing the ENTER button. Successful acceptance of the new record interval is indicated by the fact that the time of the measurement now always changes by 15 sec.



Dashboard content page

1	Time stamp of the measurement data record	2	Input field for the time between the measurement data recordings (adopted when pressing the "Enter" button)

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