



IMST GmbH

Carl-Friedrich-Gauß-Str. 2-4, D-47475 Kamp-Lintfort

Wireless M-Bus Range Extender

AN036 - Reassembling Data

Version 1.3

Document State

final

Date

November 2021

Document ID

4000/40140/0170

© 2021 IMST GmbH - All rights reserved

Revision history

Date	Version	Chapter	Description
04.02.2021	0.1	all	created
09.02.2021	1.0	all	reviewed
15.02.2021	1.1	all	refer to Package Reassembler Server Side
26.10.2021	1.2	IMST "Package Reassembler Server Side" script	HTTPS added
04.11.2021	1.3	all	typos

Content

- General Information AN036
- LoRaWAN® Network Server
 - ChirpStack
 - The Things Network Console
- IMST "Package Reassembler Server Side" script
- InfluxDB
- Grafana
- Configuration of the Wireless M-Bus Range Extender

General Information AN036

The purpose of this application note is to describe a way to interpret the uploaded LoRaWAN[®] payload at the server side.

Before going into details, a few hints about LoRaWAN.

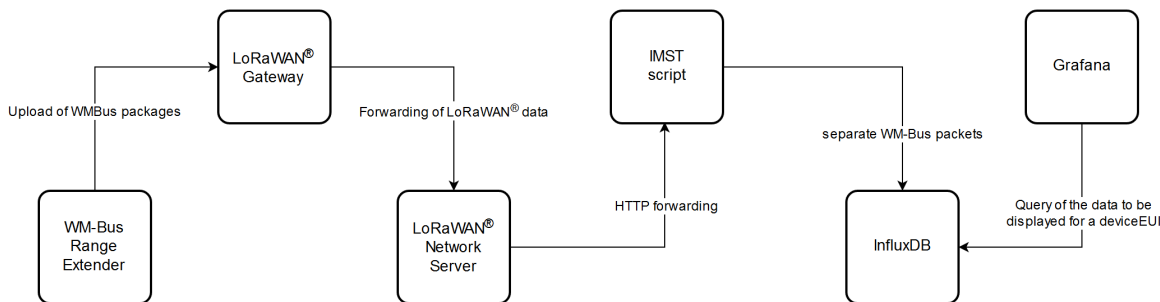
- Within LoRaWAN the possible maximum payload size of a LoRaWAN packet depends on the used data rate (spreading factor) during the transmission of the packet. Furthermore, within LoRaWAN an ADR (Adaptive Data Rate) algorithm may be used. When a packet is transmitted as a confirmed data packet an adaptation of the data rate after the transmission of the packet can appear. This change in data rate might have an immediate impact on the possible maximum payload size that can be used within a LoRaWAN packet. Due to the nature of the application data to be transmitted, it might be necessary to spread larger application data into several LoRaWAN packets. So, uploaded application data might be segmented depending on the payload length and used data rate (maximum allowed payload size of a LoRaWAN packet). Those sequences of LoRaWAN packets must be reassembled to complete application data on the receiving side again. Therefore, a tiny transport protocol is used within the device.
- For the most public available LoRaWAN[®] servers it is not possible to cache data on those servers. Therefore, the transmission of segmented data must be handled on an additional server of the user.

Due those conditions we provide these instructions on how to handle the LoRaWAN packets on server side.

IMST provides a [Node.js](#) implementation, which reassembles LoRaWAN[®] packets forwarded via HTTP from a LoRaWAN[®] Network Server and breaks them down into the Wireless M-Bus data packets. Then this data is written into an InfluxDB. The data can be easily taken from the InfluxDB and displayed with the Grafana table visualization.

Instances of the following components are required for this and have been tested with the following version numbers:

- a LoRaWAN[®] gateway
- a LoRaWAN[®] Network Server, which supports HTTP forwarding
 - ChirpStack
 - chirpstack-application-server version 3.6.1
 - chirpstack-network-server version 3.5.0
 - TTN
 - The Things Network Stack V2
- a user application server with node.js
 - node.js v8.10.0
 - npm 3.5.2
- InfluxDB
 - shell version: 1.7.9
- Grafana
 - v6.5.1



Depending on the situation, LoRaWAN[®] Network Server, IMST "Package Reassembler Server Side" script, InfluxDB and Grafana can be united on one server.

The LoRaWAN[®] Network Server must offer the possibility to forward the received data of an application to another server via the HTTP protocol. Not only the [ChirpStack](#) server, but also [The Things Network Console](#) offers this possibility.

The IMST "Package Reassembler Server Side" script is written in JavaScript and this code can be easily run with [Node.js](#). Node.js is used for running JavaScript code outside a web browser.

[Grafana](#) is an open source web application, which should be used to visualize data from InfluxDB.

Saving the data in the InfluxDB and visualizing it in Grafana should only be used as an example. After the data has been reassembled, the user should implement a custom solution on how he would like to process the data further .

Disclaimer

IMST GmbH points out that all information in this document are given on an “as is” basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication.

IMST GmbH reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete. All products are sold and delivered subject to “General Terms and Conditions” of IMST GmbH, supplied at the time of order acknowledgment.

IMST GmbH assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer's duty to bear responsibility for compliance of systems or units in which products from IMST GmbH are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications.

The product is not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the product within such applications do so at their own risk.

Any resale of IMST GmbH products or services with statements different from or beyond the parameters stated by IMST GmbH for that product/solution or service is not allowed and voids all express and any implied warranties. The limitations on liability in favor of IMST GmbH shall also affect its employees, executive personnel and bodies in the same way. IMST GmbH is not responsible or liable for any such wrong statements.

LoRaWAN® Network Server

The LoRaWAN® Network Server must offer the possibility to forward the received data of an application to another server via the HTTP protocol. Not only the [ChirpStack](#) server, but also [The Things Network Console](#) offers this possibility.

ChirpStack is an open-source LoRaWAN Network Server, which would run on a server, managed by the user.

TTN is a LoRaWAN® network server platform. It offers an open network of LoRaWAN gateways and an open source concept to manage devices and applications.

Handling of both servers is briefly described in the following chapters.

ChirpStack

If ChirpStack is used, it has to be installed on an user's server: <https://www.chirpstack.io/project/>

An own gateway has to be set up. The [IMST Lite Gateway](#) is recommended as hardware.

Instructions for registering a gateway at a Chipstack server are available at <https://www.chirpstack.io/project/guides/connect-gateway/>

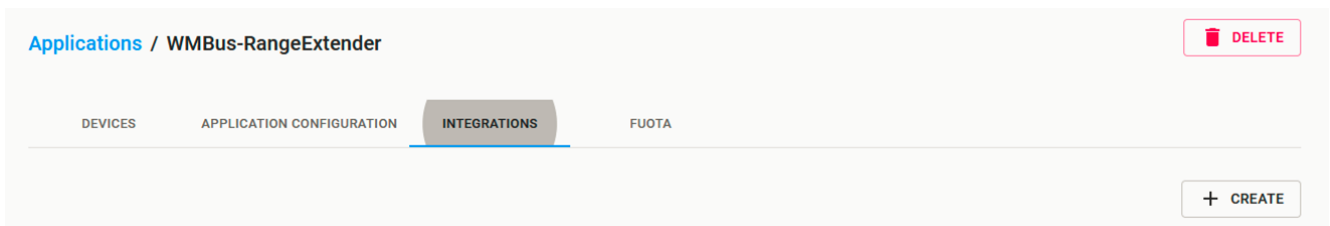
Before an end device can be created in ChirpStack, a corresponding device-profile and application must first be added. The end device has to be registered subsequent. An explanation for that can be found at <https://www.chirpstack.io/project/guides/connect-device/>

You have to [configure](#) the Wireless M-Bus Range Extender accordingly for ChirpStack (app eui & keys).

Finally a [HTTP integration](#) has to be created.

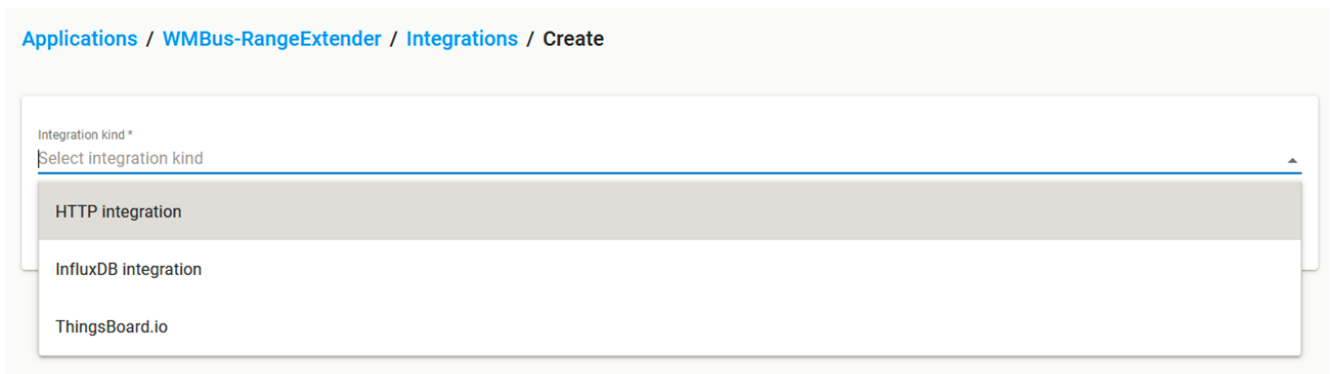
HTTP Integration

The tab integrations has to be selected in the corresponding application.



In this menu item, a new integration must then be created using the 'create' button.

HTTP integration must be selected in the newly opened menu.



The name or IP address of the user application server and a port, which can be freely selected, have to be adjusted and signed in for uplink data. The selected port will be reused for running the IMST "Package Reassembler Server Side" script.

```
http://UserApplicationServerNameOrIP:1234/WMBusRangeExtender_Uplink
```

Applications / WMBus-RangeExtender / Integrations / Create

Integration kind *

HTTP integration

Headers

ADD HEADER

Endpoints

Uplink data URL(s)

http://npmServerNameOrIP:1234/WMBusRangeExtender_Uplink

Multiple URLs can be defined as a comma separated list. Whitespace will be automatically removed.

The entered data is accepted by pressing the 'CREATE INTEGRATION' button.

Finally, the received uplink data of the corresponding application is forwarded to the IMST script on the user application server.

The Things Network Console

For using TTN, an account must be created at first. That can be done at <https://account.thethingsnetwork.org/register>.

The [overview page](#) can be used to find out whether a gateway is available in the area of the Wireless M-Bus Range Extender. If no gateway is available nearby, an own one should be set up. The [IMST Lite Gateway](#) is recommended as hardware.

Instructions for registering a gateway at TTN are available at <https://www.thethingsnetwork.org/docs/gateways/registration.html>.

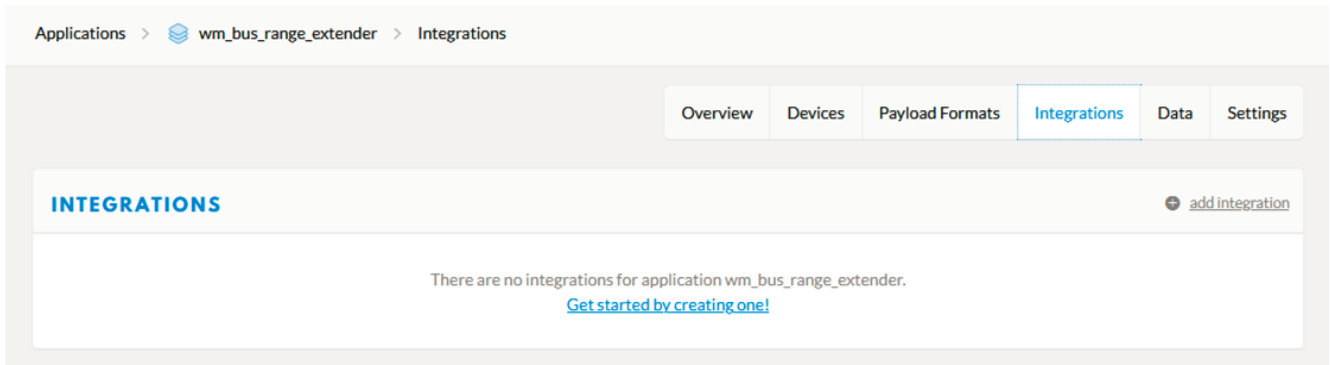
Before an end device can be created in TTN, the corresponding application must first be added, if not yet available. An explanation for that can be found at <https://www.thethingsnetwork.org/docs/applications/add.html>.

The end device has to be registered and instructions for that can be found at: <https://www.thethingsnetwork.org/docs/devices/registration.html>.

You have to [configure](#) the Wireless M-Bus Range Extender accordingly for TTN (app eui & keys).

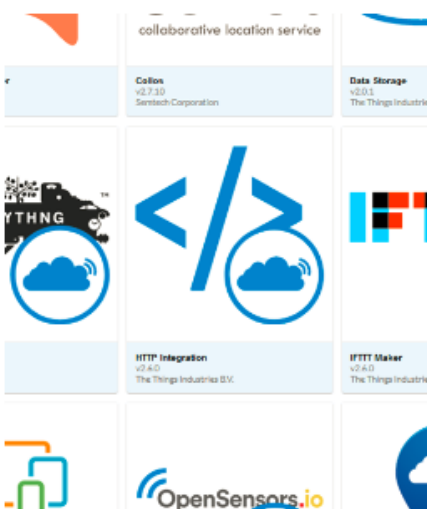
HTTP Integration

The tab *'integrations'* has to be selected in the corresponding application.



In this menu item, a new integration must then be created using the *'add integration'* button.

'HTTP Integration' must be selected in the newly opened menu.



The following input must be entered for adding a new HTTP integration:

- a name for the integration in the field *'Process ID'*
- the *'Access Key'* should be selected from the dropdown menu to *'default key'*
- the URL of the endpoint consists of the following parts
 - name or IP address of the user application server, where the IMST script is running
 - a port, which can be freely selected
 - the selected port will be reused for running the IMST script on the user application server

- and the path '*WMBusRangeExtender_Uplink*' for the script
- '*POST*' is already entered as HTTP method and should be retained

`http://UserApplicationServerNameOrIP:1234/WMBusRangeExtender_Uplink`

The screenshot shows a web interface for adding an integration. The title is 'ADD INTEGRATION' and the subtitle is 'HTTP Integration'. Below the title, there is a description: 'Make alert data to an endpoint and receive immediate HTTP response'. The form contains several fields:

- Process ID:** A text input field containing 'wmbusrangeextender_0001'. A green checkmark is visible to the right.
- Alert Key:** A text input field with a dropdown arrow to its right.
- URL:** A text input field containing 'http://UserApplicationServerNameOrIP:1234/WMBusRangeExtender_Uplink'. This field is highlighted with a blue selection bar.
- Method:** A dropdown menu set to 'POST'. A green checkmark is visible to the right.
- Authentication:** A text input field with a green checkmark to its right.
- Custom Header Name:** A text input field with a green checkmark to its right.
- Custom Header Value:** A text input field with a green checkmark to its right.

 At the bottom right of the form, there is a green button labeled 'Add integration'.

The entered data is accepted by pressing the '*Add integration*' button.

Finally, the received uplink data of the corresponding application is forwarded to the IMST script on the user application server.

IMST "Package Reassembler Server Side" script

It is mandatory to install [node.js](#), which includes npm.

The IMST script has to be extracted to a directory on the user application server.

The script is designed to handle data from Wireless M-Bus Range Extender as well as from the [IMST iO881A](#), which is an optical reader unit for smart meter.

Different HTTP Uplink data URLs are used to distinguish between those device types. These can be modified in the file 'app.js'.

```
//forward route for iO881A uplink data
app.use('/iO881A_Uplink', iO881A_Uplink);
//forward route for WM-Bus Range Extender uplink data
app.use('/WMBusRangeExtender_Uplink', WMBusRangeExtender_Uplink);
```

Adjustment on IMST Script

Security

HTTPS can be used to communicate over a secure channel with the script. For this the certificate and the private key must be provided. The file 'www' in the directory 'bin' includes the following code:

```
var privateKey = fs.readFileSync('path_to_private_key_file', 'utf8');
var certificate = fs.readFileSync('path_to_certificate_file', 'utf8');
var credentials = {key: privateKey, cert: certificate};
```

The path to the certification file has to be specified for 'path_to_certificate_file' and the associated file with the private key has to be specified for 'path_to_private_key_file'.

InfluxDB

The script is designed in such a way, that it writes the received and separated Wireless M-Bus data packages into an InfluxDB database. According to the InfluxDB settings the JavaScript code must be adjusted. The file 'Reassembling.js' in the directory 'controller' includes the following code:

```
const influx = new Influx.InfluxDB({
  host: 'InfluxDBServerNameOrIP',
  password: 'InfluxDBPassword',
  username: 'InfluxDBUsername'
})
```

The server name or IP address, on which the InfluxDB is running, must be specified for 'host'. Username and password must be specified according to a user who has access to the specified database.

Run IMST script

The command

```
npm install
```

has to be executed in the directory with a console tool so that the required packages for the project can be loaded and installed. In this context the folder 'node_modules' should be created in the working directory.

For windows systems the command

```
set PORT=1234 HTTPS_PORT=4321 & npm start
```

can be used to start the script in the working directory in a simple way.

For Linux systems the command

```
PORT=1234 HTTPS_PORT=4321 npm run start
```

should be executed.

The port number (in our example 1234 for HTTP and 4321 for HTTPS) can be freely selected, but it must match the value entered for the [HTTP forwarding on a LoRaWAN server](#).

InfluxDB

InfluxDB is a database management system and must also be installed (<https://docs.influxdata.com/influxdb/v1.7/introduction/>) on a server.

To use the script with a running InfluxDB instance, the user has to create a database in advance. At first, start the command line interface with:

```
influx -username InfluxDBUsername -password InfluxDBPassword
```

Afterwards, the new database '*WMBusDB*' should be created with the following command:

```
CREATE DATABASE WMBusDB
```

The newly created database functions as a container for all data packages created by the script at runtime. All privileges are granted to the specific user with following command:

```
GRANT ALL ON WMBusDB TO InfluxDBUsername
```

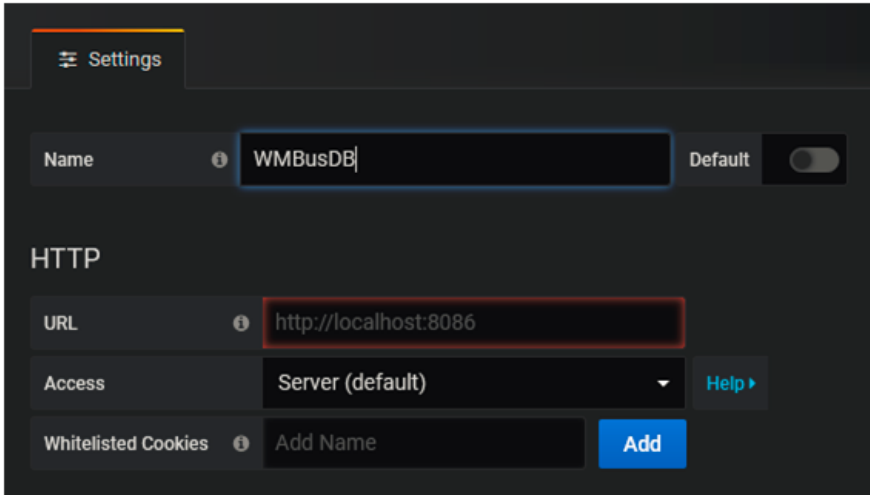
Grafana

Grafana is a tool to visualize the data and must also be installed (<https://grafana.com/docs/grafana/latest/installation/>) on a server.

InfluxDB Connection

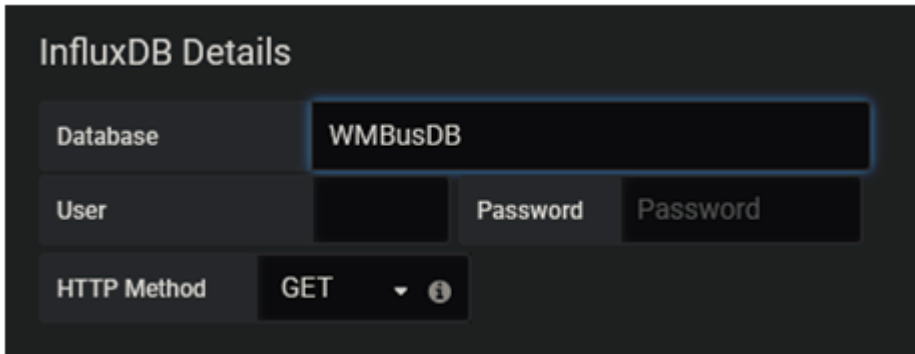
Before the created database can be used, a link must be established to Grafana. More information on how to do this is available at: <https://grafana.com/docs/grafana/latest/datasources/influxdb/>.

For easy use with our predefined dashboard, the name 'WMBusDB' should be assigned. The name or IP address of the InfluxDB server has to be entered, too.



The screenshot shows the Grafana 'Settings' page for an InfluxDB data source. The 'Name' field is set to 'WMBusDB' and the 'Default' toggle is turned off. Under the 'HTTP' section, the 'URL' field is set to 'http://localhost:8086'. The 'Access' dropdown is set to 'Server (default)'. At the bottom, there is a 'Whitelisted Cookies' section with an 'Add Name' input field and an 'Add' button.

The database 'WMBusDB' and the access data has also to be entered.



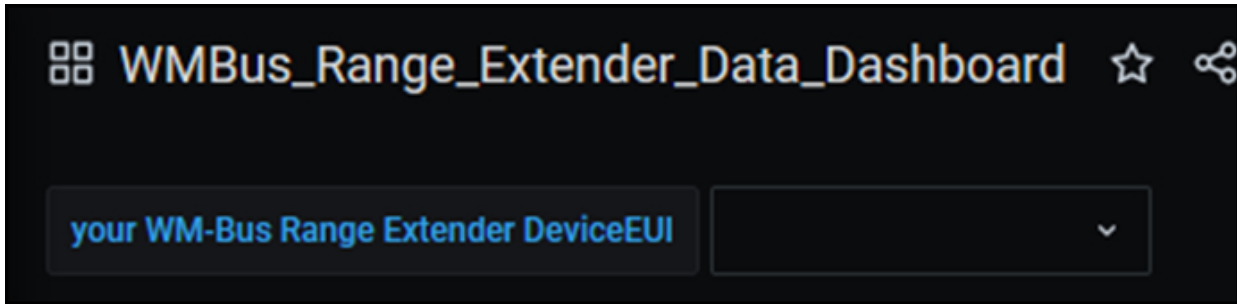
The screenshot shows the 'InfluxDB Details' configuration page. The 'Database' field is set to 'WMBusDB'. There are two 'Password' fields, one for 'User' and one for 'Password'. The 'HTTP Method' dropdown is set to 'GET'.

Dashboards

Two predefined dashboard JSON models are available:

- WMBus_Range_Extender_Data_Dashboard
- WMBus_Range_Extender_Status_Dashboard

For each dashboard, the user can choose which device should be displayed from the already available Device EUIs in the corresponding database.



The data dashboard shows the received packets in a table with the following table columns:

- Time
- WMBus.RSSI
- WMBus.Length Field
- WMBus.CTRL Field
- WMBus.Man ID
- WMBus.Device ID
- WMBus.Version
- WMBus.Type
- WMBus.WMBus Data

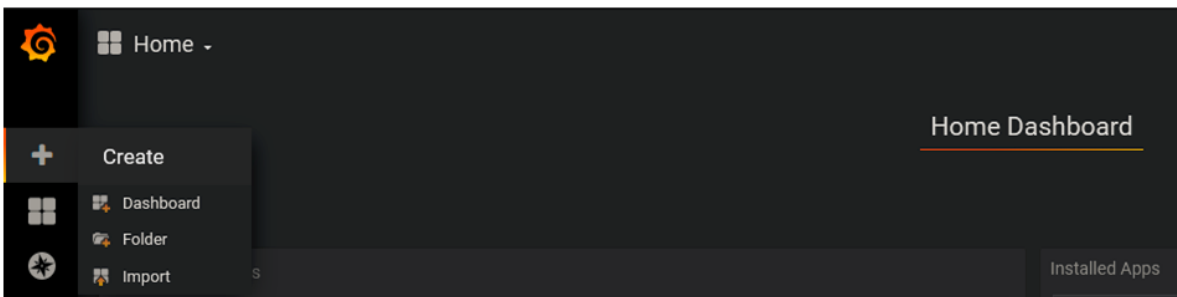
The status dashboard visualizes the following values:

- Status
 - System Time
 - Time of last Synchronization
 - Reset Counter
 - Firmware Version
 - Firmware Type
 - Battery Voltage
- State / Error
 - LoRaWAN@ Activation State
 - Network Time Synchronization State
 - System Time Synchronization State
 - LoRaWAN@ Configuration State
 - WM-Bus Address Filter List Configuration State
 - Calendar Event List Configuration State
 - Flash Memory Full State
 - Flash Memory CRC Error State
- Reader Counters
 - Number of received packets
 - Number of filtered and recorded packets
 - Number of uploaded packets

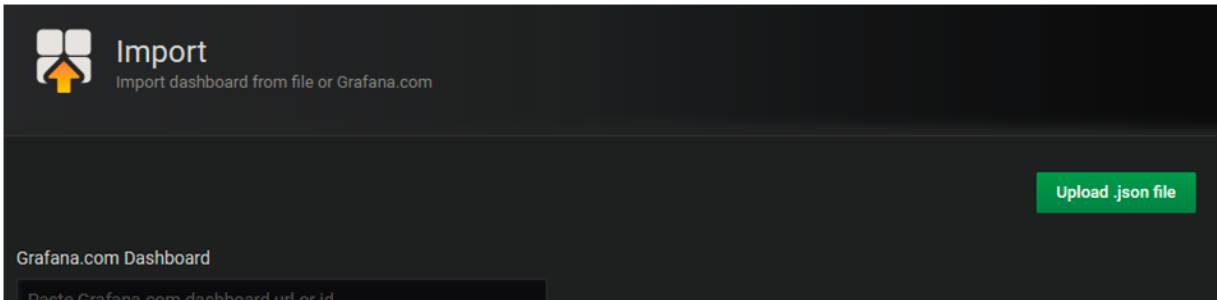
These dashboards can easily be imported into Grafana.

Import Dashboard

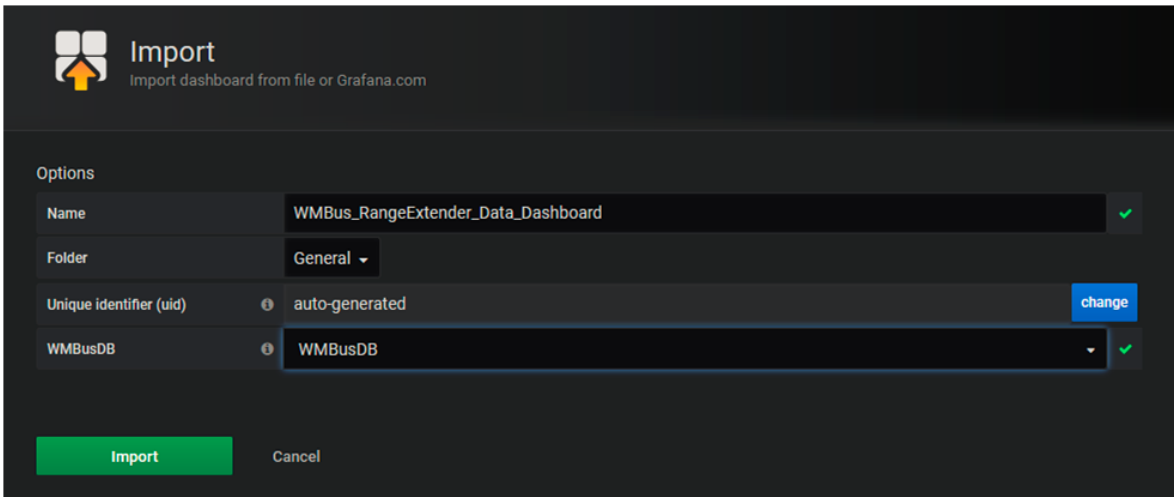
To import a dashboard the corresponding menu item must be selected via the menu.



The dashboard JSON file has to be uploaded via the corresponding button.



In the following dialog, the database, which was connected to Grafana, must be selected. The dashboard name can also be changed here.



If the Device EUI is known and there is already data in the InfluxDB, data will be displayed according to the selected period.

Time	WMBus_CTRL.Field	WMBus_Man.ID	WMBus_Device.ID	WMBus_Version	WMBus_Type	WMBus_WMBus.Data
2021-02-04 13:18:14	0x43	11-22	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f
2021-02-04 13:18:12	0x44	ff-ff	ff-ff-ff-ff	0xff	0xff	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 13:18:10	0x44	00-00	00-00-00-00	0x00	0x00	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 13:18:08	0x44	11-22	33-44-55-66	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 12:03:01	0x43	11-22	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f
2021-02-04 12:02:58	0x44	ff-ff	ff-ff-ff-ff	0xff	0xff	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 12:02:56	0x44	00-00	00-00-00-00	0x00	0x00	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 12:02:52	0x44	11-22	33-44-55-66	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 07:00:05	0x43	11-22	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f
2021-02-04 07:00:00	0x44	66-22	33-44-55-66	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:59:58	0x44	55-22	33-44-55-66	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:59:56	0x44	22-11	66-55-44-33	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:59:53	0x44	ff-ff	ff-ff-ff-ff	0xff	0xff	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:59:51	0x44	00-00	00-00-00-00	0x00	0x00	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:59:47	0x44	11-22	33-44-55-66	0x77	0x88	2a-00-00-00-2f-2f-0c-14-27-04-85-02-04-6d-32-37-1f-15-02-6d-17-00-00
2021-02-04 06:54:01	0x43	25-b3	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f
2021-02-04 06:53:57	0x43	b3-25	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f
2021-02-04 06:53:54	0x43	11-22	33-44-55-66	0x77	0x88	01-02-03-04-05-06-07-08-09-0a-0b-0c-0d-0e-0f-10-11-12-13-14-15-16-17-18-19-1a-1b-1c-1d-1e-1f

Each *panel* can be subsequently changed and adapted to the user needs by using the *panel editor*.

Configuration of the Wireless M-Bus Range Extender

The following documents are recommended for adjusting settings of the Wireless M-Bus Range Extender:

- WMBus_Range_Extender_AN032_QuickStartGuide
- WMBus_Range_Extender_UserManual
- WSConfigurator_UserManual_Range_Extender

The Wireless M-Bus Range Extender has to be configured according to the LoRaWAN[®] server settings relating the selected activation type and must be activated. A repeated event to receiving Wireless M-Bus packages and to upload the stored Wireless M-Bus packages and/or to send the status should be defined as calendar event.

Once the device has been set up and is connected to the LoRaWAN[®] network and communicates with the LoRaWAN[®] server, the device data should be displayed in Grafana, if everything is configured correctly.