IMST GmbH

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

iOKE868 LoRaWAN®

AN034 - Reassembling Data

Version 1.2

Document State

final

Date

April 2023

Document ID

4000/40140/0167

Revision history

Date	Version	Chapter	Description
18.12.2020	0.1	all	created
14.01.2021	1.0	all	reviewed
04.02.2021	1.1	all	little corrections
11.04.2023	1.2	all	adjustments

Content

General Information AN034

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.

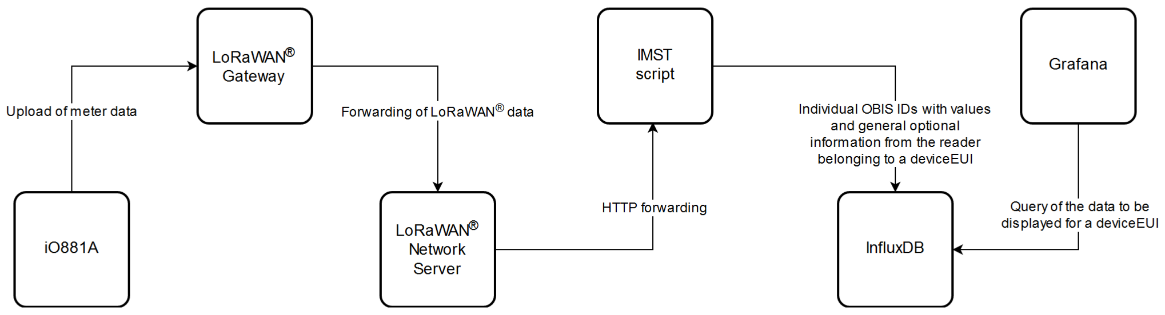
- Within LoRaWAN the possible maximum payload size of a LoRaWAN packet depends on the used data rate (spreading factor) during the transmission of the packet. Furthermore, within LoRaWAN an ADR (Adaptive Data Rate) algorithm is used. When a packet is transmitted as a confirmed data packet an adaptation 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 packtes 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 this 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 their individual values. Then these values are written into an InfluxDB. The data can be easily taken from the InfluxDB and displayed with the Grafana visualization.

Instances of the following components are required for this:

- a LoRaWAN® gateway
- a LoRaWAN® Network Server, which supports HTTP forwarding
- a user application server with node.js
- InfluxDB
- Grafana


LoRaWAN® Gateway
IMST script
Grafana
Upload of meter data
Forwarding of LoRaWAN® data
Individual OBIS IDs with values and general optional information from the reader belonging to a deviceEUI
HTTP forwarding
Query of the data to be displayed for a deviceEUI
iO881A
LoRaWAN® Network Server
InfluxDB

Depending on the situation, LoRaWAN® Network Server, IMST 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 offer this possibility.

The IMST script is written in JavaScript and this code can be easily run with Node.js. Node.js is used for runing JavaScript code outside a web browser.

Grafana is an open source web application, which should be used to visualize data from InfluxDB.

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.

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 discussed in the following chapters.

ChirpStack

Reassembelling the data was developed and tested with Chirpstack v3.

If ChirpStack is used, it has to be installed on a users 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/

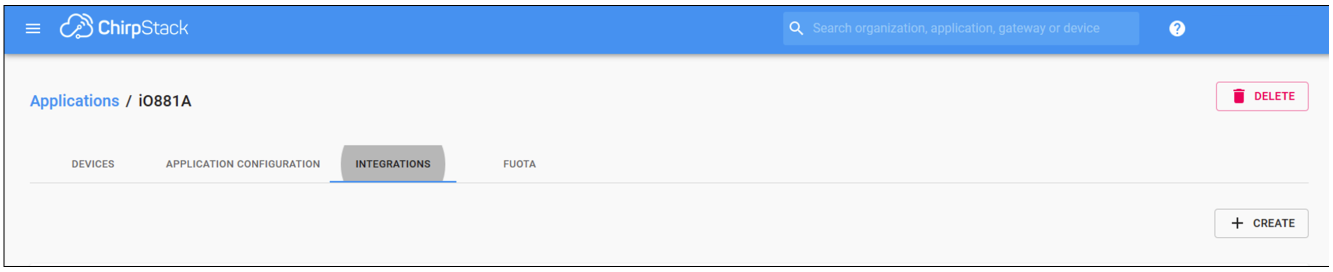
Before an end device can be created in ChirpStack, a corresponding device-profile and application must first be added. The end device has to be registered subsequent. An explanation for that can be found at https://www.chirpstack.io/project/guides/connect-device/

You have to configure the iO881A accordingly for ChirpStack (app eui & keys).

Finally a HTTP integration has to be created.

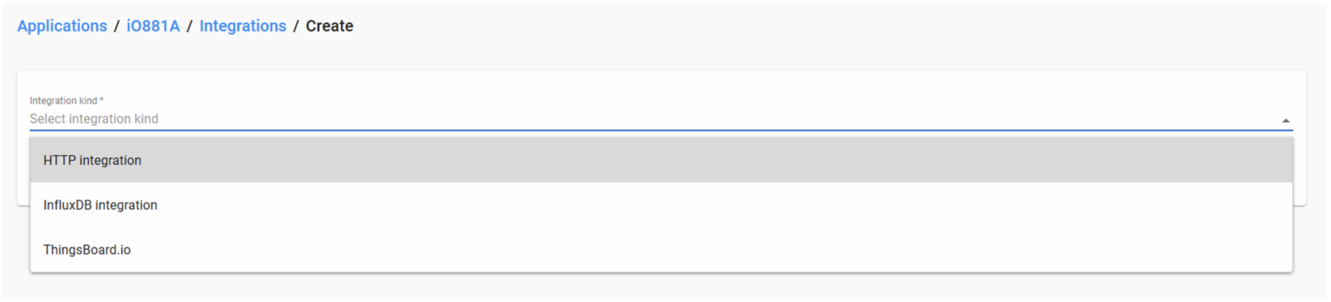
HTTP Integration

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


ChirpStack
Search organization, application, gateway or device
Applications / iO881A
DELETE
DEVICES
APPLICATION CONFIGURATION
INTEGRATIONS
FUOTA
CREATE

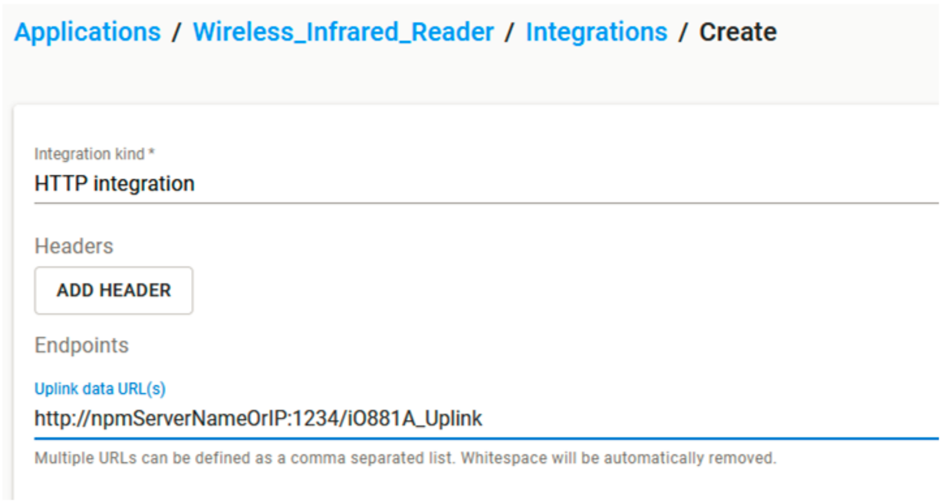
In this menu item, a new integration must then be created using the '*create*' button.

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 script.

```
http://UserApplicationServerNameOrIP:1234/iO881A_Uplink
```


Applications / Wireless_Infrared_Reader / Integrations / Create

Integration kind *
HTTP integration

Headers

Endpoints

Uplink data URL(s)
http://npmServerNameOrIP:1234/iO881A_Uplink

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

The entered data are 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://www.thethingsnetwork.org/get-started.

The overview page can be used to find out whether a gateway is available in the area of the iO881A. 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.

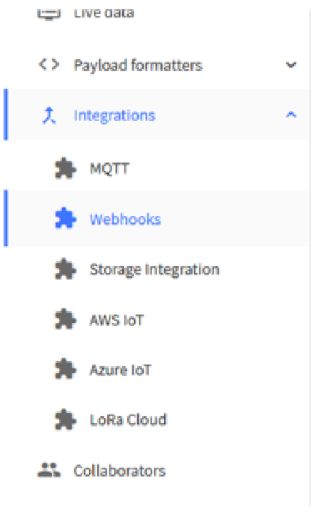
Before an end device can be created in TTN, the corresponding application must first be added, if not yet available. An explanation for that can be found at https://www.thethingsnetwork.org/docs/applications/add.html.

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 iO881A accordingly for TTN (app eui & keys).

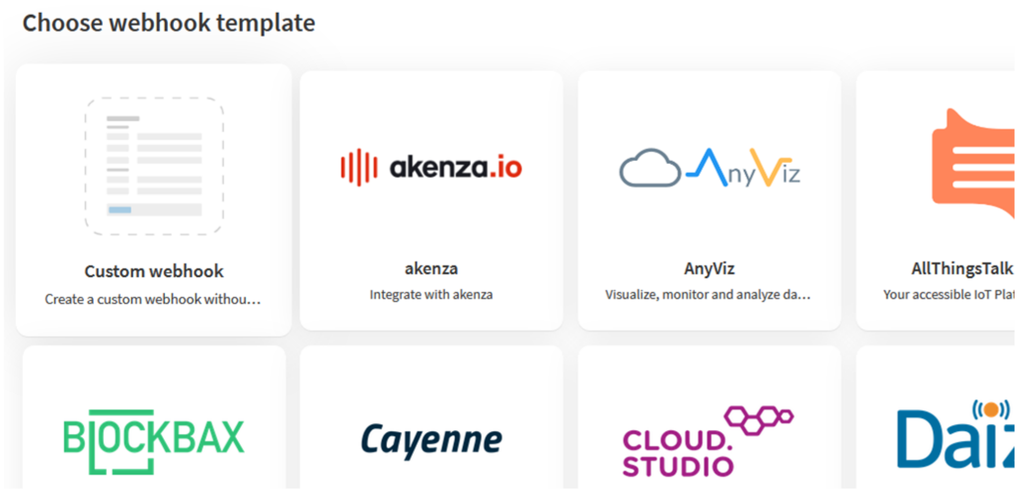
HTTP Integration

The menu item '*Integrations* / Webhooks' has to be selected in the corresponding application.



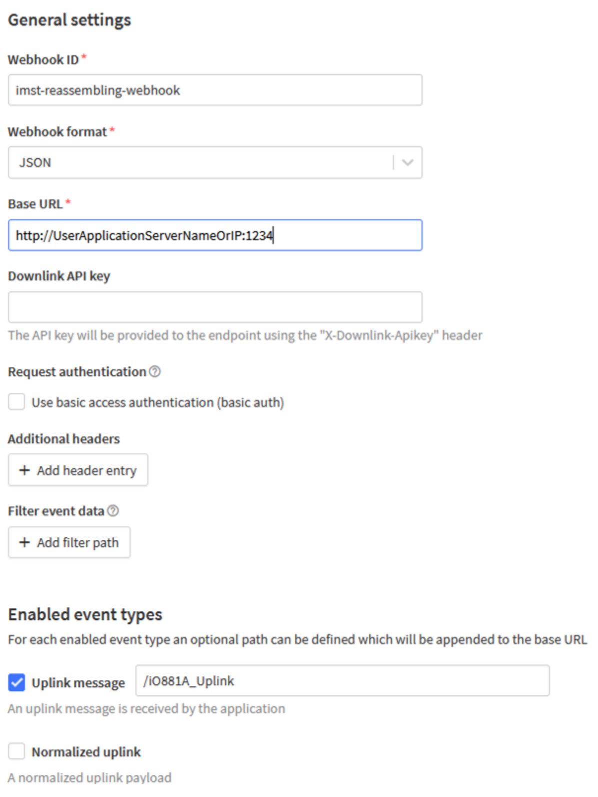
In this menu item, a new integration must then be created using the '*add webhook*' button.

'*Custom webhook*' must be selected in the newly opened menu.



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

- a name for the webhook in the field '*Webhook ID*'
- 'JSON' is already entered as '*Webhook format*' method and should be retained
- the '*Access Key*' should be selected from the dropdown menu to '*default key*'
- the '*Base URL*' of the endpoint consists of the following parts
  - the protocol *http* or *https*
  - 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 '*iO881A_Uplink*' for the script
- '*Uplink message*' has to be enabled in the '*Enabled event types*' and the path '*iO881A_Uplink*' for the script ha to be entered


General settings

Webhook ID *
imst-reassembling-webhook

Webhook format *
JSON

Base URL *
http://UserApplicationServerNameOrIP:1234

Downlink API key

The API key will be provided to the endpoint using the "X-Downlink-Apikey" header

Request authentication
☐ Use basic access authentication (basic auth)

Additional headers
+ Add header entry

Filter event data
+ Add filter path

Enabled event types
For each enabled event type an optional path can be defined which will be appended to the base URL

☑ Uplink message /iO881A_Uplink
An uplink message is received by the application

☐ Normalized uplink
A normalized uplink payload

The entered data are accepted by pressing the '*Add webhook*' button.

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

IMST 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 iO881A as well as from the IMST Wireless M-Bus Range Extender, which is device that collects wireless M-Bus messages from utility meters and forwards them to a LoRaWAN® network.

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 for InfluxDB

The script is designed in such a way, that it writes the received OBIS value 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 & npm start
```

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

For Linux systems the command

```
PORT=1234 npm run start
```

should be executed.

The port number (in our example 1234) can be freely selected, but it must match the value entered for the HTTP(S) 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.

Reassembelling the data was developed and tested with InfluxDB Version 1.8.10.

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 '*iO881A_DB*' should be created with the following command:

```
CREATE DATABASE iO881A_DB
```

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

```
GRANT ALL ON iO881A_DB TO InfluxDBUsername
```

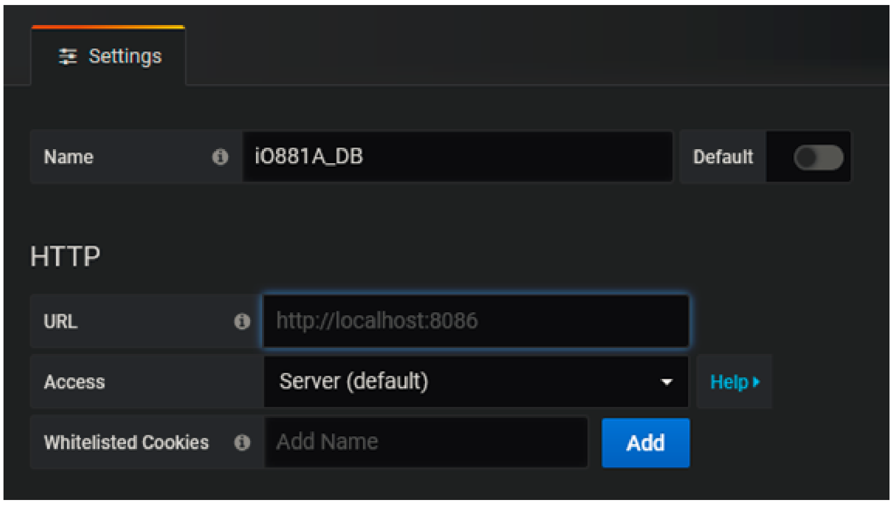
Grafana

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 dashboards, the name '*iO881A_DB*' should be assigned. The name or IP address of the InfluxDB server has to be entered, too.



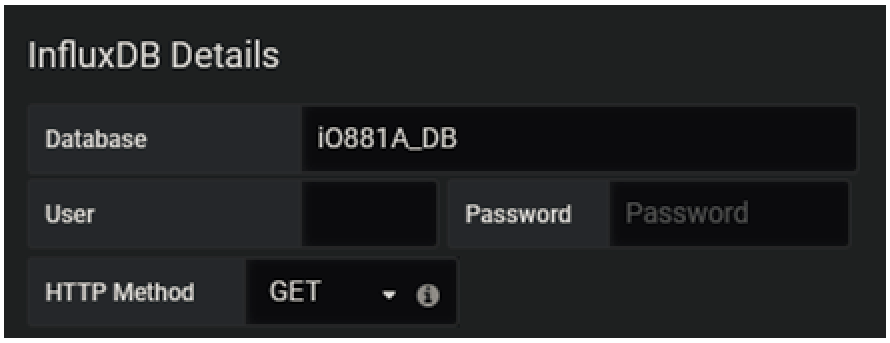
Name: iO881A_DB
Default

HTTP

URL: http://localhost:8086
Access: Server (default)

Whitelisted Cookies: Add Name

The database '*iO881A_DB*' and the access data has also to be entered.


InfluxDB Details

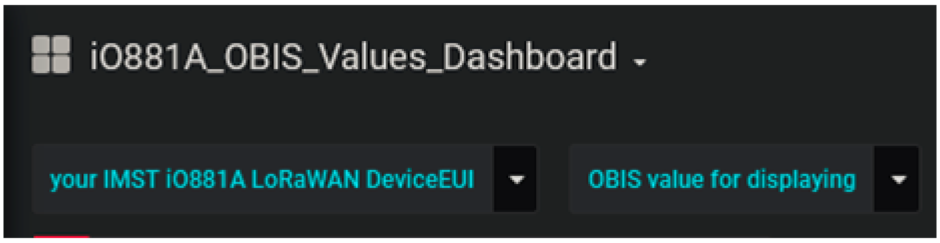
Database: iO881A_DB
User:
Password: Password
HTTP Method: GET

Dashboards

Two predefined dashboard JSON models are available:

- iO881A_OBIS_Values_Dashboard
- iO881A_Status_Dashboard

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


iO881A_OBIS_Values_Dashboard

your IMST iO881A LoRaWAN DeviceEUI
OBIS value for displaying

The OBIS values dashboard has the following panel to OBIS IDs relationships:

Manufacturer ID	129-129-199-130-3-255
Ownership ID	1-0-0-0-0-255
Chart	selectable data is only displayed if the value of the selected obis id is numeric
Table	selectable

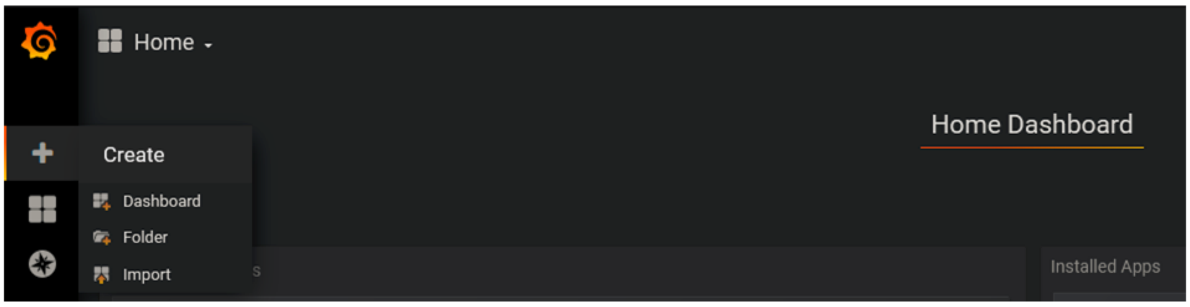
The status dashboard visualizes the following values:

- Status
    - iO881A System Time
    - Time of last Synchronization
    - Firmware Version
    - Reset Counter
- State / Error
    - LoRaWAN® Activation State
    - Network Time Synchronization State
    - System Time Synchronization State
    - Over The Air Activation Procedure State
    - LoRaWAN® Configuration State
    - Calendar Event List Configuration State
    - OBIS ID Filter List Configuration State
- Reader Counters
    - Number of correctly received meter files
    - Number of faulty received meter files with read / CRC errors
    - Number of uploaded meter data messages

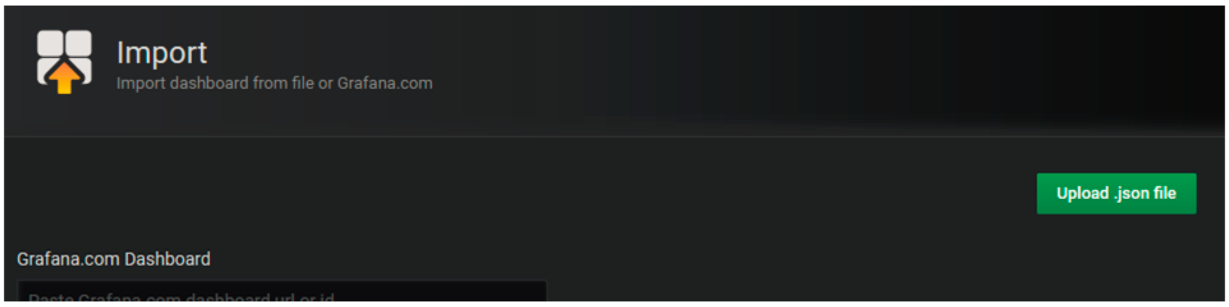
These dashboards can easily be imported into Grafana.

Import Dashboard

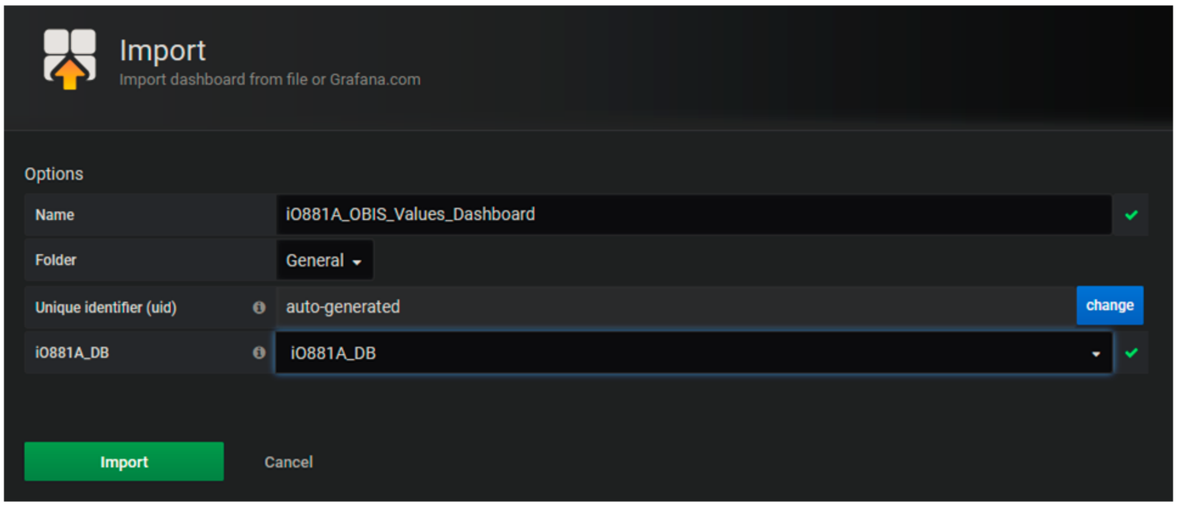
To import a dashboard the corresponding menu item must be selected via the menu.



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.



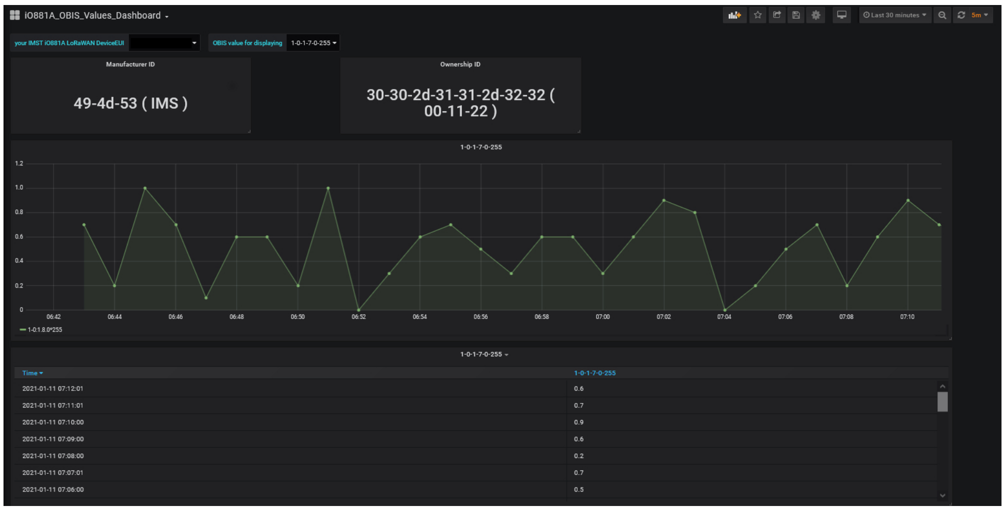
Import

Import dashboard from file or Grafana.com

Options

Name	iO881A_OBIS_Values_Dashboard
Folder	General
Unique identifier (uid)	auto-generated
iO881A_DB	iO881A_DB

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



iO881A_OBIS_Values_Dashboard

your IMST iO881A LoRaWAN DeviceEUI: [redacted]

OBIS value for displaying: 1-0-1-7-0-255

Manufacturer ID: 49-4d-53 (IMS)

Ownership ID: 30-30-2d-31-31-2d-32-32 (00-11-22)

1-0-1-7-0-255

Time	1-0-1-7-0-255
2021-01-11 07:12:01	0.6
2021-01-11 07:11:01	0.7
2021-01-11 07:10:00	0.9
2021-01-11 07:09:00	0.6
2021-01-11 07:08:00	0.2
2021-01-11 07:07:01	0.7
2021-01-11 07:06:00	0.5

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

Configuration of the iO881A

The following documents are recommended for adjusting settings of the iO881A:

- iOKE868_LoRaWAN_AN031_QuickStartGuide
- iOKE868_LoRaWAN_UserManual
- WSConfigurator_UserManual_iO881A

The iO881A has to be configured according to the LoRaWAN® server settings relating the selected activation type and must be activated. A repeated event to read out at least one OBIS Id 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.