Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Department of the Environment, Transport, Energy and Communications DETEC

Swiss Federal Office of Energy SFOE
Energy Research and Cleantech Division

REEL Demo – Romande Energie ELectric network in local balance Demonstrator

Deliverable: 3a2 First results of control of DSM Units by GridEye

Demo site: Chapelle

Developed by
Peter Gysel, FHNW
Michael Roth, FHNW
Dominik Link, FHNW
in collaboration with
DEPsys, Romande Energie
[Olten, 28.12.2019]

1. Description of deliverable and goal

1.1. Executive summary

The goal of this milestone was to test DSM functionality. One of the key elements was to retrieve measurement data from nodes placed in the electrical grid in Rolle provided by GridEye. This task could be implemented successfully but should be improved and tested further after the recommendations we sent to DEPsys are implemented. There are also some open points which are under discussion with the Romande Energie IT team. After those are set, all the use cases regarding data access can be tested.

The second part, controlling devices for showing DSM, could not be implemented as the partners for the DSM have changed multiple times in the last years and the situation has not been steady enough to implement such features.

Additionally, we implemented a proxy to retrieve measurement data programmatically through a secure gateway which enables us to provide fine granular permissions on a variety of data sources combined in a single endpoint.

1.2. Research question

How can data from partners such as DEPsys be saved in a central database. And how can it be retrieved from various users?

1.3. Novelty of the proposed solutions compared to the state-of-art

A custom python3 script was written for pulling data from the DEPsys API. This script is stateless and is invokes through a reoccurring cron task which is set up on the central server. The script transforms data obtained from the DEPsys API and inserts the resulting data set into influxdb. While this process is custom-tailored to the demonstrator and its environment, it is in fact state-of-the art, therefore it is mere engineering. However, the task required a lot of discussions and re-designing as the whole ecosystem of the demonstrator was changing a lot.

1.4. Description

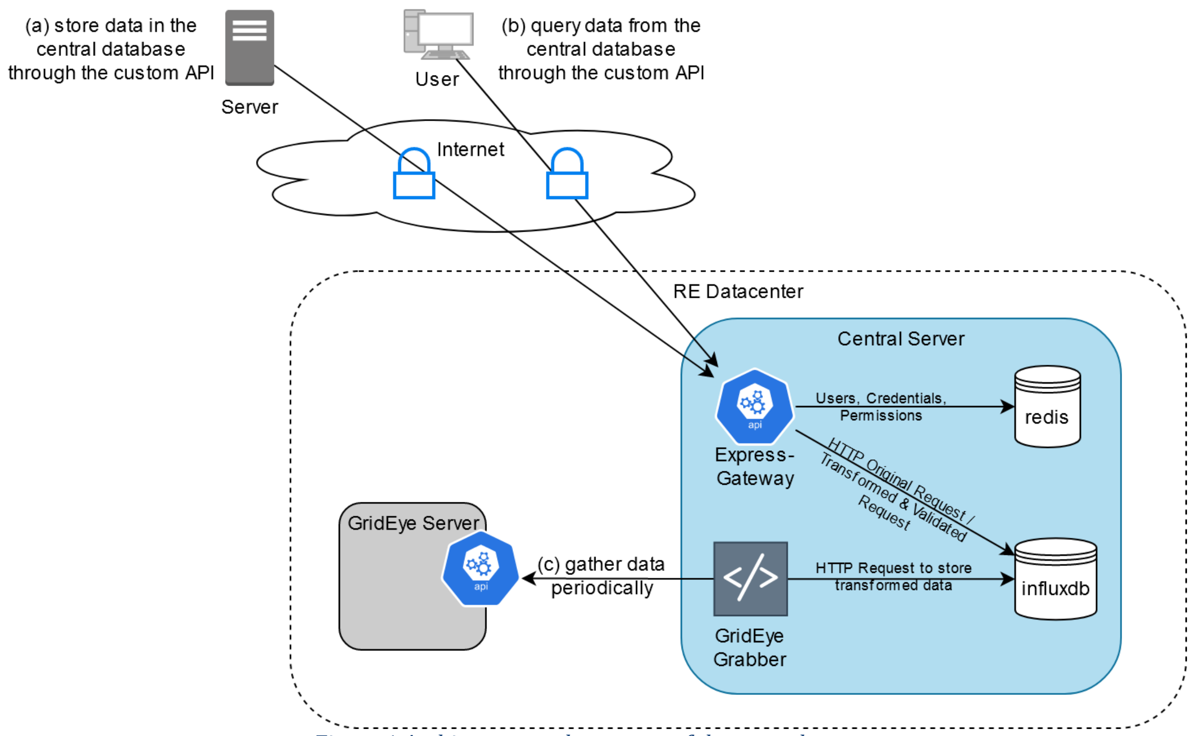
The central server was moved to its destination in the Romande Energie datacenter. After a complete reinstallation, the core components (Express Gateway + redis and influxdb) were installed and configured as described in the previous milestone 'WP3.4

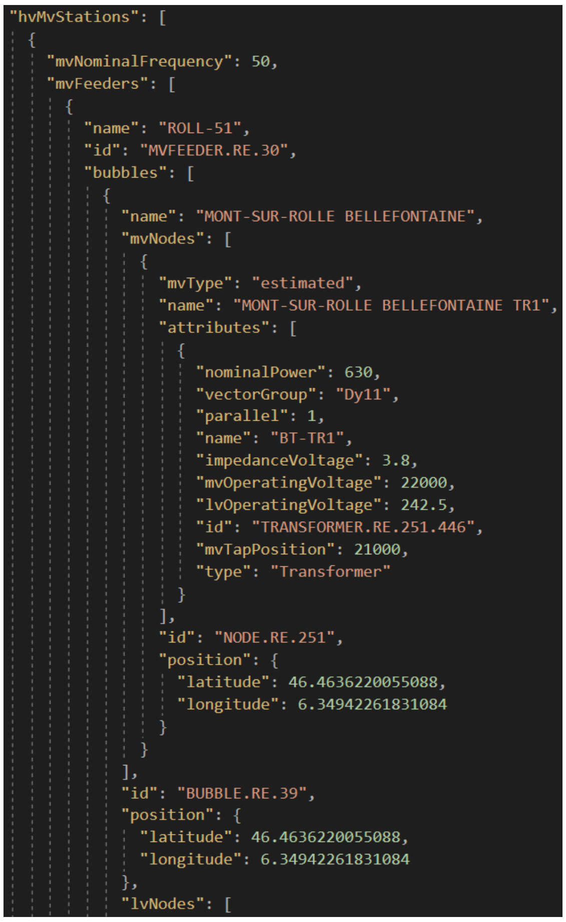
Communication specification GridEye and DSM'. The Express Gateway serves as a custom interface to influxdb (the central database) which handles custom authentication, authorization and possible data validation and data transformation. This enables the use cases (a) and (b) (see Figure 1) where third-parties either store or retrieve data data through this API.

In addition to finalizing this installation, the major use case (c) was implemented. Data from the DEPsys GridEye server is queried periodically in a python3 script. This script is stateless and is executed periodically using cron. Python was chosen as it is a wide-spread and relatively simple programming language. Therefore, this script can be adjusted rapidly by anyone who has minor programming experience.

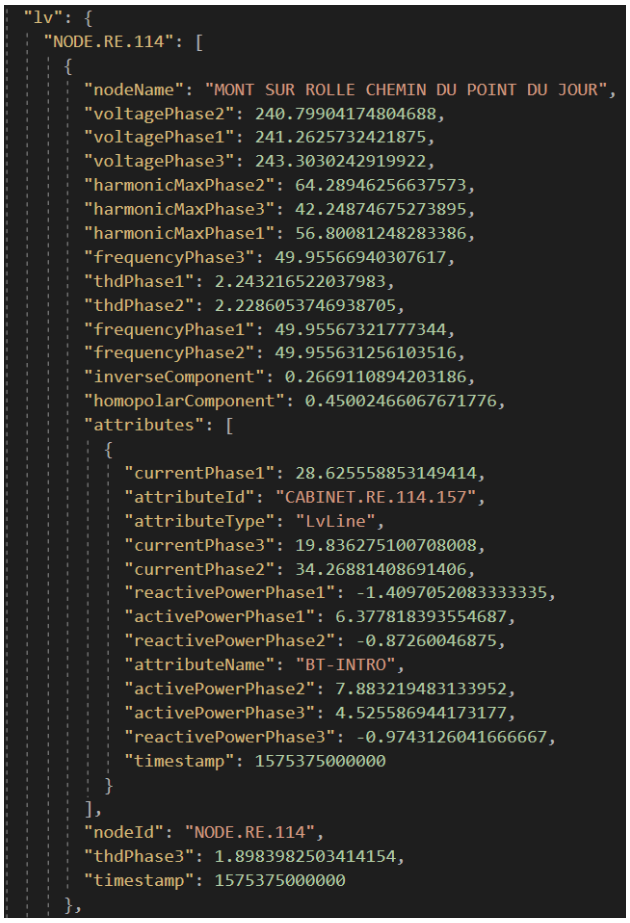
This script starts by querying the data structure of the GridEye API. This data structure contains necessary information such as which measuring nodes are available and which *bubble* (container data structure) they are placed in. After parsing this data structure, the script has the information necessary to query the measurements.

The second part queries all the bubbles to retrieve measurement data from the last 24 hours. There are several API calls to achieve this as one query must be sent for each bubble. The answer JSON is then processed. Certain information such as the relationship to the bubble have to be inserted into each measurement. In addition, there are attributes (see Figure 2) inside such a measurement with an own timestamp. The fields of such a data entry are then stored either as metadata (key-value-pairs) for indexing and querying or measured data values as seen in the example in Figure 3. As seen in Figure 2, the incoming data sets have an additional hierarchy 'attributes. Those are also measurements which are stored separately using another influx-measurement (something like a table in a relational database).


(a) store data in the central database through the custom API
Server
User
(b) query data from the central database through the custom API
Internet
RE Datacenter
Central Server
api
Express-Gateway
Users, Credentials, Permissions
redis
HTTP Original Request / Transformed & Validated Request
GridEye Server
api
(c) gather data periodically
GridEye Grabber
HTTP Request to store transformed data
influxdb
Figure 1 Architecture and use cases of the central server

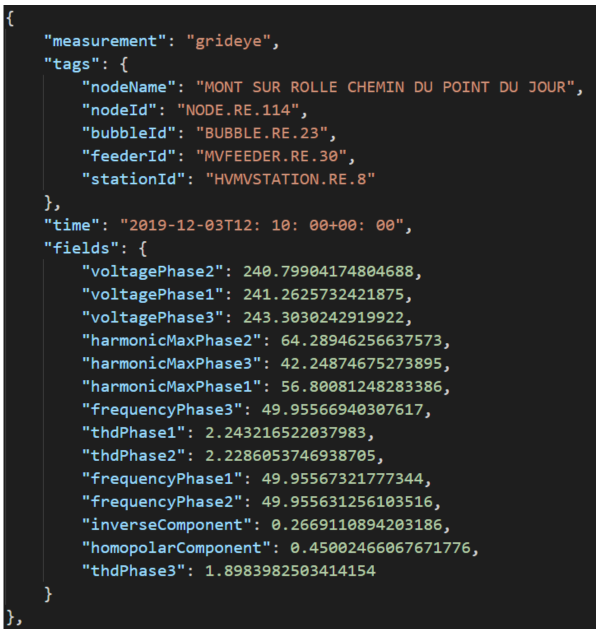


```
"hvMvStations": [
  {
    "mvNominalFrequency": 50,
    "mvFeeders": [
      {
        "name": "ROLL-51",
        "id": "MVFEEDER.RE.30",
        "bubbles": [
          {
            "name": "MONT-SUR-ROLLE BELLEFONTAINE",
            "mvNodes": [
              {
                "mvType": "estimated",
                "name": "MONT-SUR-ROLLE BELLEFONTAINE TR1",
                "attributes": [
                  {
                    "nominalPower": 630,
                    "vectorGroup": "Dy11",
                    "parallel": 1,
                    "name": "BT-TR1",
                    "impedanceVoltage": 3.8,
                    "mvOperatingVoltage": 22000,
                    "lvOperatingVoltage": 242.5,
                    "id": "TRANSFORMER.RE.251.446",
                    "mvTapPosition": 21000,
                    "type": "Transformer"
                  }
                ],
                "id": "NODE.RE.251",
                "position": {
                  "latitude": 46.4636220055088,
                  "longitude": 6.34942261831084
                }
              }
            ],
            "id": "BUBBLE.RE.39",
            "position": {
              "latitude": 46.4636220055088,
              "longitude": 6.34942261831084
            },
            "lvNodes": [
```

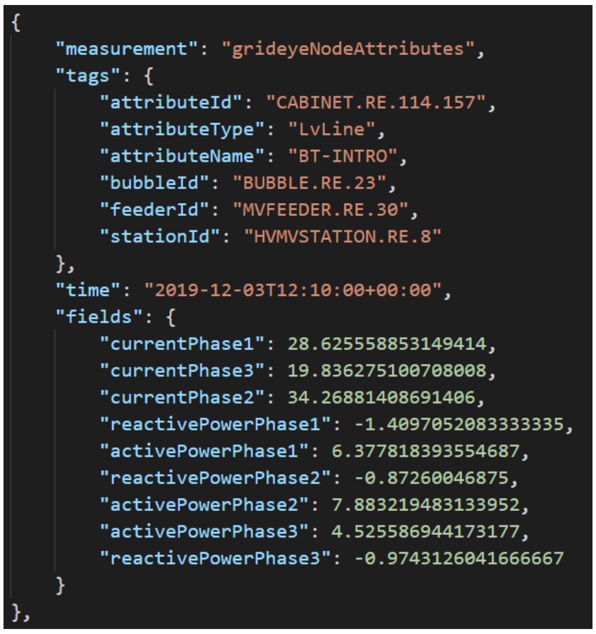


```
"lv": {
  "NODE.RE.114": [
    {
      "nodeName": "MONT SUR ROLLE CHEMIN DU POINT DU JOUR",
      "voltagePhase2": 240.79904174804688,
      "voltagePhase1": 241.2625732421875,
      "voltagePhase3": 243.3030242919922,
      "harmonicMaxPhase2": 64.28946256637573,
      "harmonicMaxPhase3": 42.24874675273895,
      "harmonicMaxPhase1": 56.80081248283386,
      "frequencyPhase3": 49.95566940307617,
      "thdPhase1": 2.243216522037983,
      "thdPhase2": 2.2286053746938705,
      "frequencyPhase1": 49.95567321777344,
      "frequencyPhase2": 49.955631256103516,
      "inverseComponent": 0.2669110894203186,
      "homopolarComponent": 0.45002466067671776,
      "attributes": [
        {
          "currentPhase1": 28.625558853149414,
          "attributeId": "CABINET.RE.114.157",
          "attributeType": "LvLine",
          "currentPhase3": 19.836275100708008,
          "currentPhase2": 34.26881408691406,
          "reactivePowerPhase1": -1.4097052083333335,
          "activePowerPhase1": 6.377818393554687,
          "reactivePowerPhase2": -0.87260046875,
          "attributeName": "BT-INTRO",
          "activePowerPhase2": 7.883219483133952,
          "activePowerPhase3": 4.525586944173177,
          "reactivePowerPhase3": -0.9743126041666667,
          "timestamp": 1575375000000
        }
      ],
      "nodeId": "NODE.RE.114",
      "thdPhase3": 1.8983982503414154,
      "timestamp": 1575375000000
    },
```

Figure 2 Structure and measured Data from the API



```
{
    "measurement": "grideye",
    "tags": {
        "nodeName": "MONT SUR ROLLE CHEMIN DU POINT DU JOUR",
        "nodeId": "NODE.RE.114",
        "bubbleId": "BUBBLE.RE.23",
        "feederId": "MVFEEDER.RE.30",
        "stationId": "HVMVSTATION.RE.8"
    },
    "time": "2019-12-03T12: 10: 00+00: 00",
    "fields": {
        "voltagePhase2": 240.79904174804688,
        "voltagePhase1": 241.2625732421875,
        "voltagePhase3": 243.3030242919922,
        "harmonicMaxPhase2": 64.28946256637573,
        "harmonicMaxPhase3": 42.24874675273895,
        "harmonicMaxPhase1": 56.80081248283386,
        "frequencyPhase3": 49.95566940307617,
        "thdPhase1": 2.243216522037983,
        "thdPhase2": 2.2286053746938705,
        "frequencyPhase1": 49.95567321777344,
        "frequencyPhase2": 49.955631256103516,
        "inverseComponent": 0.2669110894203186,
        "homopolarComponent": 0.45002466067671776,
        "thdPhase3": 1.8983982503414154
    }
},
```



```
{
    "measurement": "grideyeNodeAttributes",
    "tags": {
        "attributeId": "CABINET.RE.114.157",
        "attributeType": "LvLine",
        "attributeName": "BT-INTRO",
        "bubbleId": "BUBBLE.RE.23",
        "feederId": "MVFEEDER.RE.30",
        "stationId": "HVMVSTATION.RE.8"
    },
    "time": "2019-12-03T12:10:00+00:00",
    "fields": {
        "currentPhase1": 28.625558853149414,
        "currentPhase3": 19.836275100708008,
        "currentPhase2": 34.26881408691406,
        "reactivePowerPhase1": -1.4097052083333335,
        "activePowerPhase1": 6.377818393554687,
        "reactivePowerPhase2": -0.87260046875,
        "activePowerPhase2": 7.883219483133952,
        "activePowerPhase3": 4.525586944173177,
        "reactivePowerPhase3": -0.9743126041666667
    }
},
```

Figure 3 Transformed data which is inserted into the database

2. Achievement of deliverable:

2.1. Date

20. Dec 2019

2.2. Demonstration of the deliverable

Gathering data from the GridEye API (use-case (c) in Figure 1) was implemented successfully on this side. The script gathers data from the said API, transforms data and inserts it into the database as described in chapter 1.4.
We reported an error which occurs when querying certain data on the GridEye API. We also asked if they could investigate slow API responses of approximately 60-120 minutes as this could indicate some performance issues in their backend. We also found possible anomalies in the data which we reported to DEPsys.

As server certificates and port forwardings are still under discussion with the Romande Energie IT team, the use cases (a) and (b) in Figure 1 are implemented on the server side but could only be tested on local test machines.

3. Impact

This milestone builds up on 'WP3.4 Communication specification GridEye and DSM'. The achievement of both milestones result in an application which is able to do the following:

- Periodically retrieve data from DEPsys' GridEye API for devices in Rolle
- Transforming this data from the original format into a format suitable for the centralized influxdb database.
- Allowing authorized third-parties to retrieve this data through an API which allows custom authentication and authorization.

With this implemented and running, we show that it is possible to attract data from various sources by either pulling data through a script (as implemented for this milestone) or receiving data through the API which exposes parts of the central API.