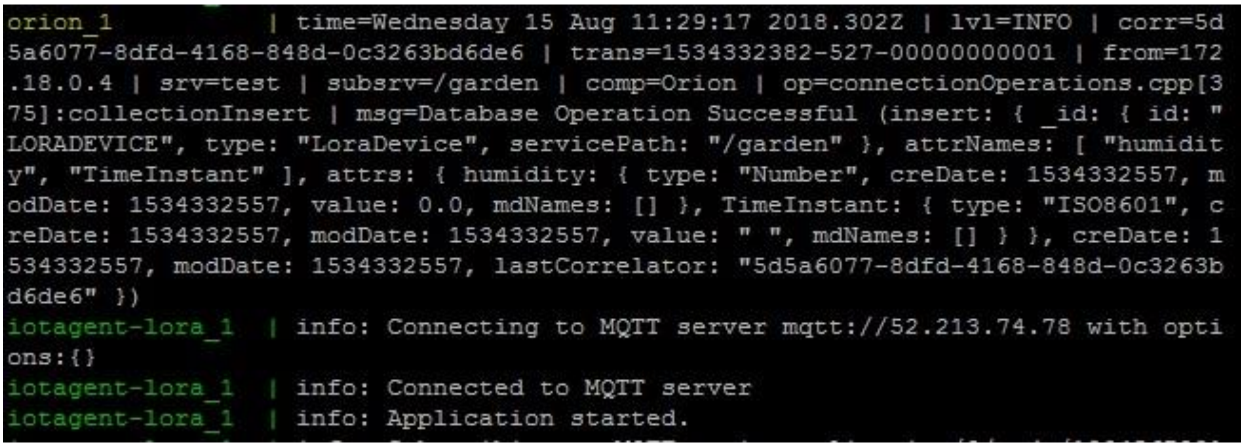
Fiware

As already mentioned before, fiware is an existent platform that has an available lab for direct experimentation. But using it that way won't allow us to make it accessible by other external users. So, we had to install its components in a server. We had a local one owned by SUP'COM, another one owned by ooredoo and we also installed it in an AWS ubuntu instance that we made.

After a lot of documentation and experimentation, we decided to use a project made by Fiware's partner ATOS that we imported from : https://github.com/Atos-Research-and-Innovation/IoTagent-LoRaWAN

The next step is to configure each component's bind address and port and simply launch them using docker-compose. The following figure shows the result of the launching



```
orion_1            | time=Wednesday 15 Aug 11:29:17 2018.302Z | lvl=INFO | corr=5d
5a6077-8dfd-4168-848d-0c3263bd6de6 | trans=1534332382-527-00000000001 | from=172
.18.0.4 | srv=test | subsrv=/garden | comp=Orion | op=connectionOperations.cpp[3
75]:collectionInsert | msg=Database Operation Successful (insert: { _id: { id: "
LORADEVICE", type: "LoraDevice", servicePath: "/garden" }, attrNames: [ "humidit
y", "TimeInstant" ], attrs: { humidity: { type: "Number", creDate: 1534332557, m
odDate: 1534332557, value: 0.0, mdNames: [] }, TimeInstant: { type: "ISO8601", c
reDate: 1534332557, modDate: 1534332557, value: " ", mdNames: [] } }, creDate: 1
534332557, modDate: 1534332557, lastCorrelator: "5d5a6077-8dfd-4168-848d-0c3263b
d6de6" })
iotagent-lora_1    | info: Connecting to MQTT server mqtt://52.213.74.78 with opti
ons:{}
iotagent-lora_1    | info: Connected to MQTT server
iotagent-lora_1    | info: Application started.
```

LoRa node and gateway

The gateway and node were prepared by another group, But the first one was made using a raspberry pi, an Rfm95 module and and the second was made with an Rfm95 module and a dht22 sensor.

Packet forwarder

The packet forwarder that was used in this project in a 'single channel packet forwarder' that sends data using a single frequency which is 868.100000 Mhz.

LoRa network

LoRa gateway bridge

Requirements

- Mosquitto MQTT broker: In order to install it, I executed the following command: sudo apt-get install mosquitto

- PostgreSQL database: LoRa Server persists the gateway data into a PostgreSQL database. To install the latest PostgreSQL:

```
wget –quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key
add - sudo echo "deb http://apt.postgresql.org/pub/repos/apt/ jessie, trusty or xenial-
pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list sudo apt-get update sudo apt-get
install postgresql-9.6
```

- Redis database: LoRa Server stores all non-persistent data into a Redis datastore. Installation on Debian / Ubuntu: sudo apt-get install redis-server

Installation of LoRa Gateway Bridge:

It's done simply by executing the following command that creates a configuration file is located at /etc/lora-gateway-bridge/lora-gatewaybridge.toml.

```
sudo apt-get install lora-gateway-bridge
```

Configuration

- Gateway: Modify the packet-forwarder of the gateway so that it will send its data to the LoRa Gateway Bridge. serveraddress to the IP address / hostname of the LoRa Gateway Bridge servportup to 1700 (the default port that LoRa Gateway Bridge is using) servportdown to 1700 (same)

- The configuration file: entering the packet forwarder's address which is the same as gateway's.

LoRa network server

Having the same requirements as the gateway bridge, LoRa Server needs its own database.

Postgres database creation

1. Starting the prompt: sudo -u postgres psql
2. Creating a user: create role loraserverNs with login password ’dbpassword’;
3. Creating a database create database loraserverNs with owner loraserverNs;

Installing the Network Server

sudo apt-get install loraserver

LoRa application server

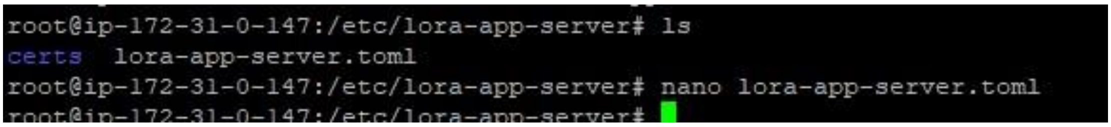
Creating a user, database and a PostgreSQL pqtrgm extension After starting the prompt we entered the following commands:

1. Creating a user: create role loraserverAs with login password ’dbpassword’;
2. Creating a database create database loraserverAs with owner loraserverAs;
3. Changing to the LoRa App Server database: loraserverAs
4. Enabling the extension: create extension pgtrgm;

Installing the Application Server

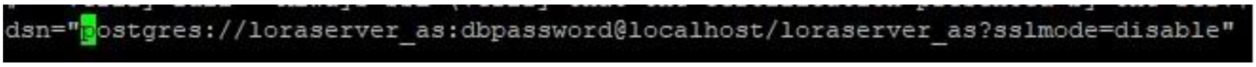
sudo apt-get install lora-app-server

After installation, modify the configuration file which is located at /etc/lora-app-server/loraapp-server.toml



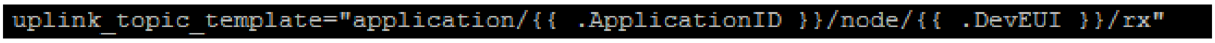
```
root@ip-172-31-0-147:/etc/lora-app-server# ls
certs  lora-app-server.toml
root@ip-172-31-0-147:/etc/lora-app-server# nano lora-app-server.toml
root@ip-172-31-0-147:/etc/lora-app-server# 
```

Given that the password that I used when creating the PostgreSQL database is ’dbpassword’, the config variable postgresql.dsn had to be changed into:



```
dsn="postgres://loraserver_as:dbpassword@localhost/loraserver_as?sslmode=disable"
```

Another thing that has to be changed in the name of the node in the uplink topic subsciption. In pour case the node is called 'node' so the code in the configuration file has to look like the screenshot below

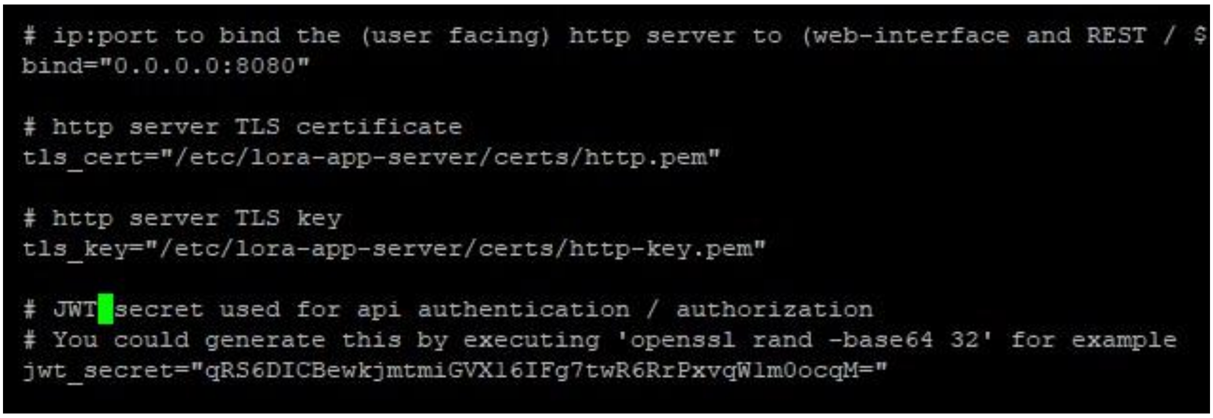


```
uplink_topic_template="application/{{ .ApplicationID }}/node/{{ .DevEUI }}/rx"
```

The application server's web interface

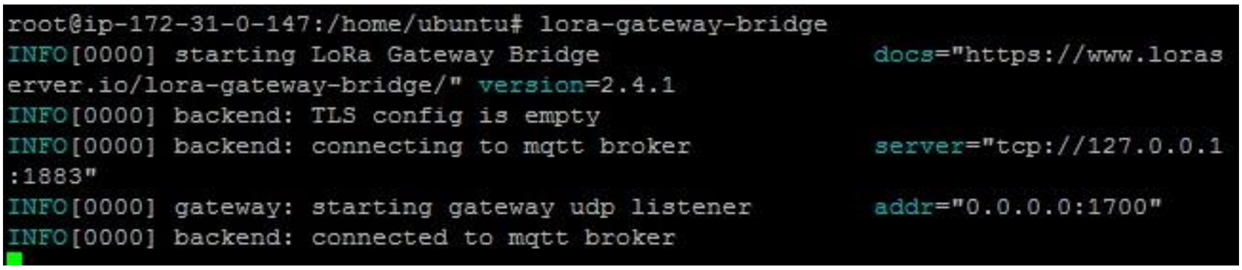
LoRa application server comes with a user-web-interface that enables the management of the different components of the network and that allows the decoding and the visualization of the received data. The following steps lead to the configuration of this interface.

1. Configuring the bind address TLS certificates and the authentication key



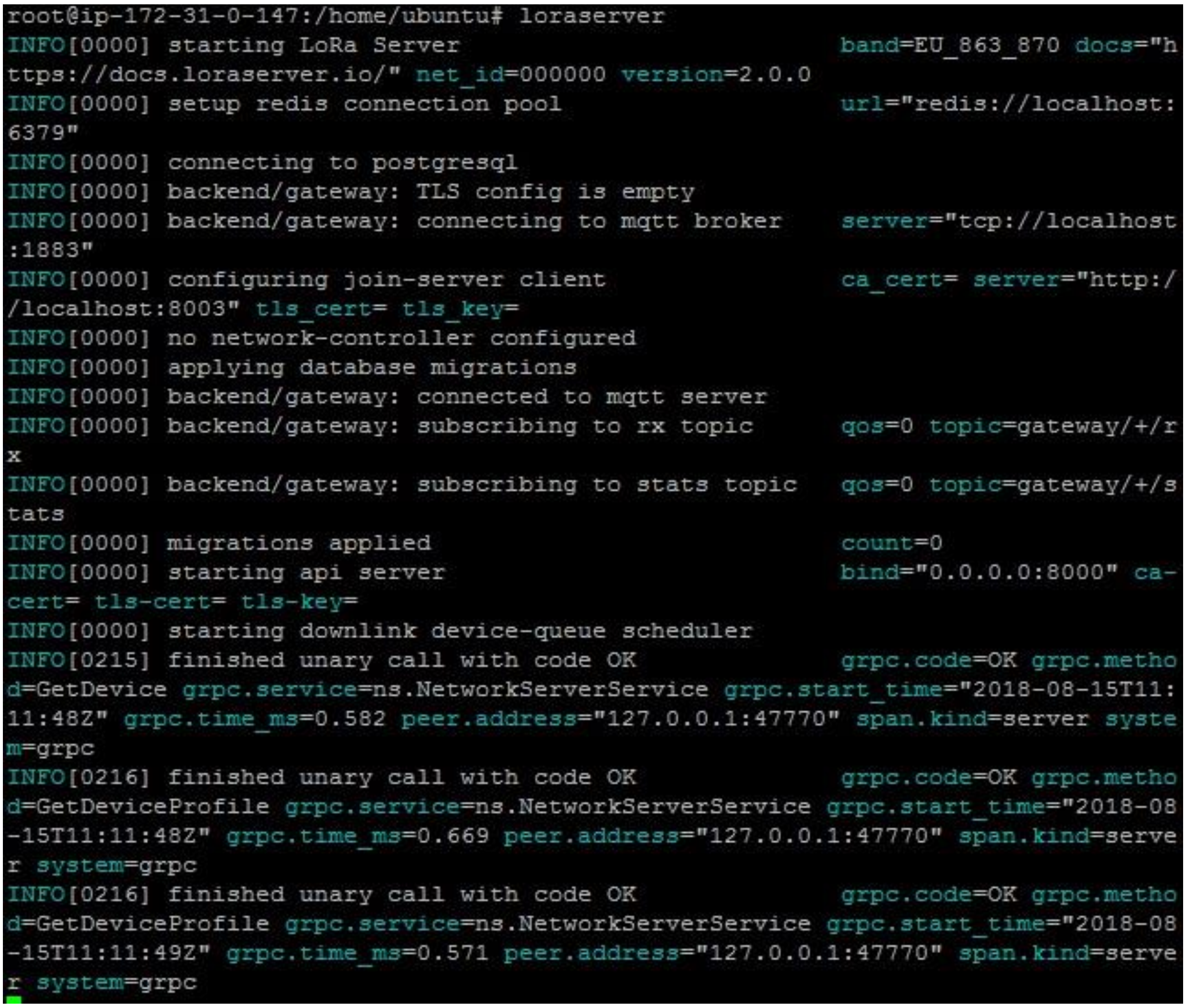
```
# ip:port to bind the (user facing) http server to (web-interface and REST / $
bind="0.0.0.0:8080"

# http server TLS certificate
tls_cert="/etc/lora-app-server/certs/http.pem"

# http server TLS key
tls_key="/etc/lora-app-server/certs/http-key.pem"

# JWT secret used for api authentication / authorization
# You could generate this by executing 'openssl rand -base64 32' for example
jwt_secret="qRS6DICBewkjmtmiGVX16IFg7twR6RrPxvqW1m0ocqM="
```

2. Launching the gateway bridge



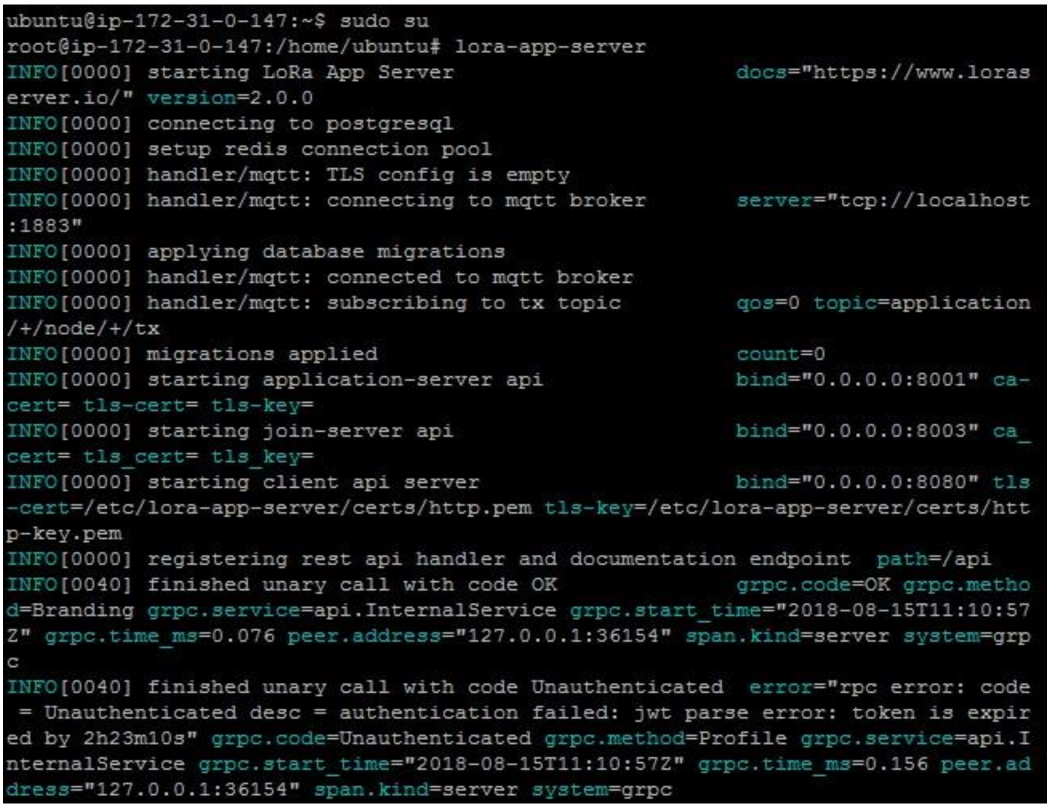
```
root@ip-172-31-0-147:/home/ubuntu# lora-gateway-bridge
INFO[0000] starting LoRa Gateway Bridge                  docs="https://www.loras
erver.io/lora-gateway-bridge/" version=2.4.1
INFO[0000] backend: TLS config is empty
INFO[0000] backend: connecting to mqtt broker            server="tcp://127.0.0.1
:1883"
INFO[0000] gateway: starting gateway udp listener        addr="0.0.0.0:1700"
INFO[0000] backend: connected to mqtt broker
```

3. Launching the loRa network server



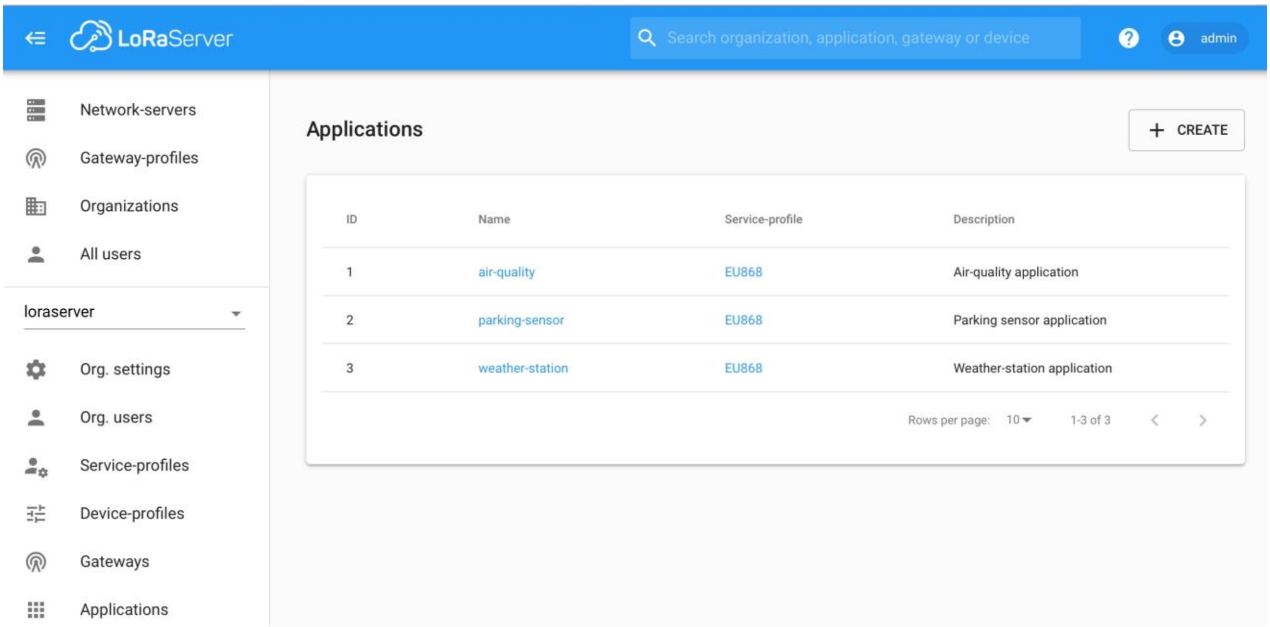
```
root@ip-172-31-0-147:/home/ubuntu# loraserver
INFO[0000] starting LoRa Server                          band=EU_863_870 docs="h
ttps://docs.loraserver.io/" net_id=000000 version=2.0.0
INFO[0000] setup redis connection pool                   url="redis://localhost:
6379"
INFO[0000] connecting to postgresql
INFO[0000] backend/gateway: TLS config is empty
INFO[0000] backend/gateway: connecting to mqtt broker    server="tcp://localhost
:1883"
INFO[0000] configuring join-server client                ca_cert= server="http:/
/localhost:8003" tls_cert= tls_key=
INFO[0000] no network-controller configured
INFO[0000] applying database migrations
INFO[0000] backend/gateway: connected to mqtt server
INFO[0000] backend/gateway: subscribing to rx topic      qos=0 topic=gateway/+/r
x
INFO[0000] backend/gateway: subscribing to stats topic   qos=0 topic=gateway/+/s
tats
INFO[0000] migrations applied                            count=0
INFO[0000] starting api server                           bind="0.0.0.0:8000" ca-
cert= tls-cert= tls-key=
INFO[0000] starting downlink device-queue scheduler
INFO[0215] finished unary call with code OK              grpc.code=OK grpc.metho
d=GetDevice grpc.service=ns.NetworkServerService grpc.start_time="2018-08-15T11:
11:48Z" grpc.time_ms=0.582 peer.address="127.0.0.1:47770" span.kind=server syste
m=grpc
INFO[0216] finished unary call with code OK              grpc.code=OK grpc.metho
d=GetDeviceProfile grpc.service=ns.NetworkServerService grpc.start_time="2018-08
-15T11:11:48Z" grpc.time_ms=0.669 peer.address="127.0.0.1:47770" span.kind=serve
r system=grpc
INFO[0216] finished unary call with code OK              grpc.code=OK grpc.metho
d=GetDeviceProfile grpc.service=ns.NetworkServerService grpc.start_time="2018-08
-15T11:11:49Z" grpc.time_ms=0.571 peer.address="127.0.0.1:47770" span.kind=serve
r system=grpc
```

4. Launching LoRa application server



```
ubuntu@ip-172-31-0-147:~$ sudo su
root@ip-172-31-0-147:/home/ubuntu# lora-app-server
INFO[0000] starting LoRa App Server                      docs="https://www.loras
erver.io/" version=2.0.0
INFO[0000] connecting to postgresql
INFO[0000] setup redis connection pool
INFO[0000] handler/mqtt: TLS config is empty
INFO[0000] handler/mqtt: connecting to mqtt broker       server="tcp://localhost
:1883"
INFO[0000] applying database migrations
INFO[0000] handler/mqtt: connected to mqtt broker
INFO[0000] handler/mqtt: subscribing to tx topic         qos=0 topic=application
/+/node/+/tx
INFO[0000] migrations applied                            count=0
INFO[0000] starting application-server api               bind="0.0.0.0:8001" ca-
cert= tls-cert= tls-key=
INFO[0000] starting join-server api                      bind="0.0.0.0:8003" ca_
cert= tls_cert= tls_key=
INFO[0000] starting client api server                    bind="0.0.0.0:8080" tls
-cert=/etc/lora-app-server/certs/http.pem tls-key=/etc/lora-app-server/certs/htt
p-key.pem
INFO[0000] registering rest api handler and documentation endpoint  path=/api
INFO[0040] finished unary call with code OK              grpc.code=OK grpc.metho
d=Branding grpc.service=api.InternalService grpc.start_time="2018-08-15T11:10:57
Z" grpc.time_ms=0.076 peer.address="127.0.0.1:36154" span.kind=server system=grp
c
INFO[0040] finished unary call with code Unauthenticated  error="rpc error: code
 = Unauthenticated desc = authentication failed: jwt parse error: token is expir
ed by 2h23m10s" grpc.code=Unauthenticated grpc.method=Profile grpc.service=api.I
nternalService grpc.start_time="2018-08-15T11:10:57Z" grpc.time_ms=0.156 peer.ad
dress="127.0.0.1:36154" span.kind=server system=grpc
```

5. Access the configured bind address, in our case it's: https://52.211.159.35:8080



Applications

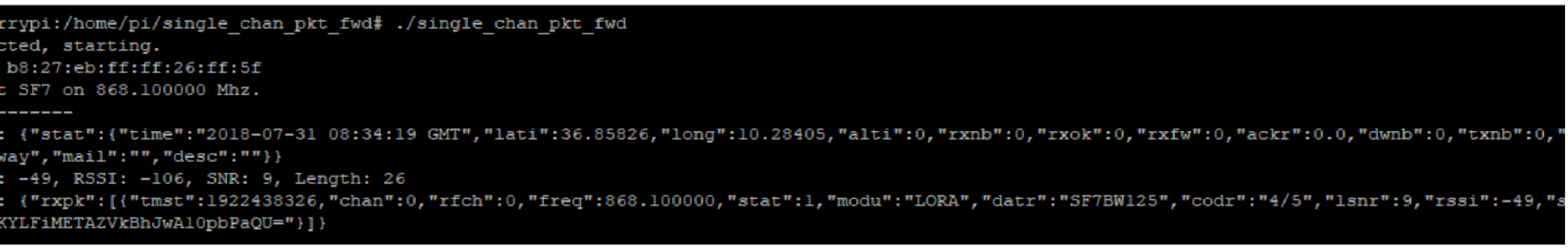
ID	Name	Service-profile	Description
1	air-quality	EU868	Air-quality application
2	parking-sensor	EU868	Parking sensor application
3	weather-station	EU868	Weather-station application

6. In there, the used gateway's as well as the device's information have to be provided, respectively in the 'gateway profiles' and 'device profiles' sections showed in the figure above. The used network server with which the application server will deal must be

entered in the ’network servers’ section. And in the ’Applications’ section, an application using the mentioned gateway and devices can be made. In the application we can choose to decode using a self developed javascript code or using CayenneLPP schema. In our case the latter one was chosen.

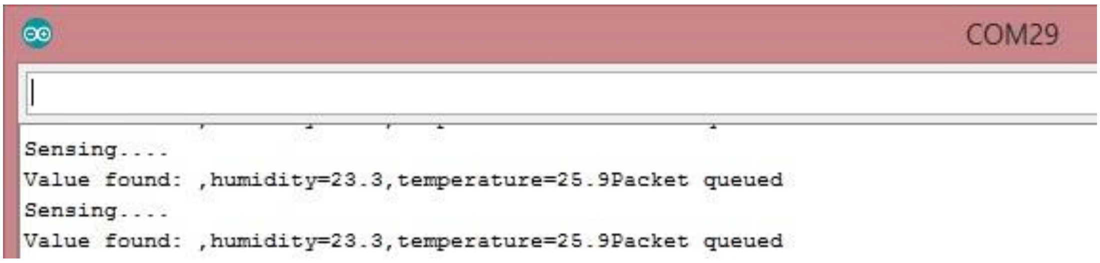
Device provisioning and data’s circulation visualization

1. Launching the packet forwarder



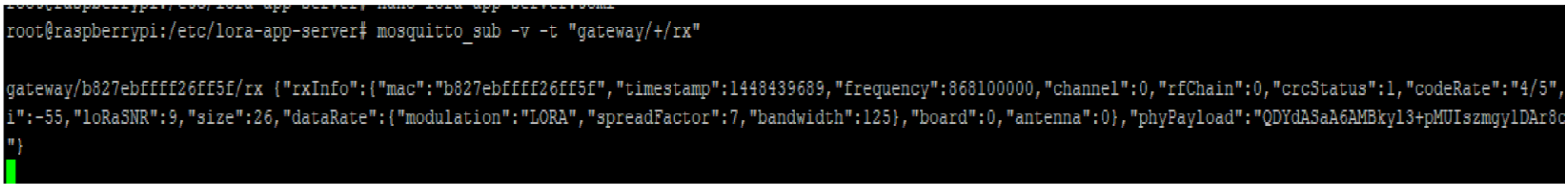
```
rrypi:/home/pi/single_chan_pkt_fwd# ./single_chan_pkt_fwd
cted, starting.
 b8:27:eb:ff:ff:26:ff:5f
t SF7 on 868.100000 Mhz.
-------
: {"stat":{"time":"2018-07-31 08:34:19 GMT","lati":36.85826,"long":10.28405,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"
way","mail":"","desc":""}}
: -49, RSSI: -106, SNR: 9, Length: 26
: {"rxpk":[{"tmst":1922438326,"chan":0,"rfch":0,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125","codr":"4/5","lsnr":9,"rssi":-49,"s
XYLFiMETAZVkBhJwA10pbPaQU="}]}
```

And here are the values that are being sent by the node



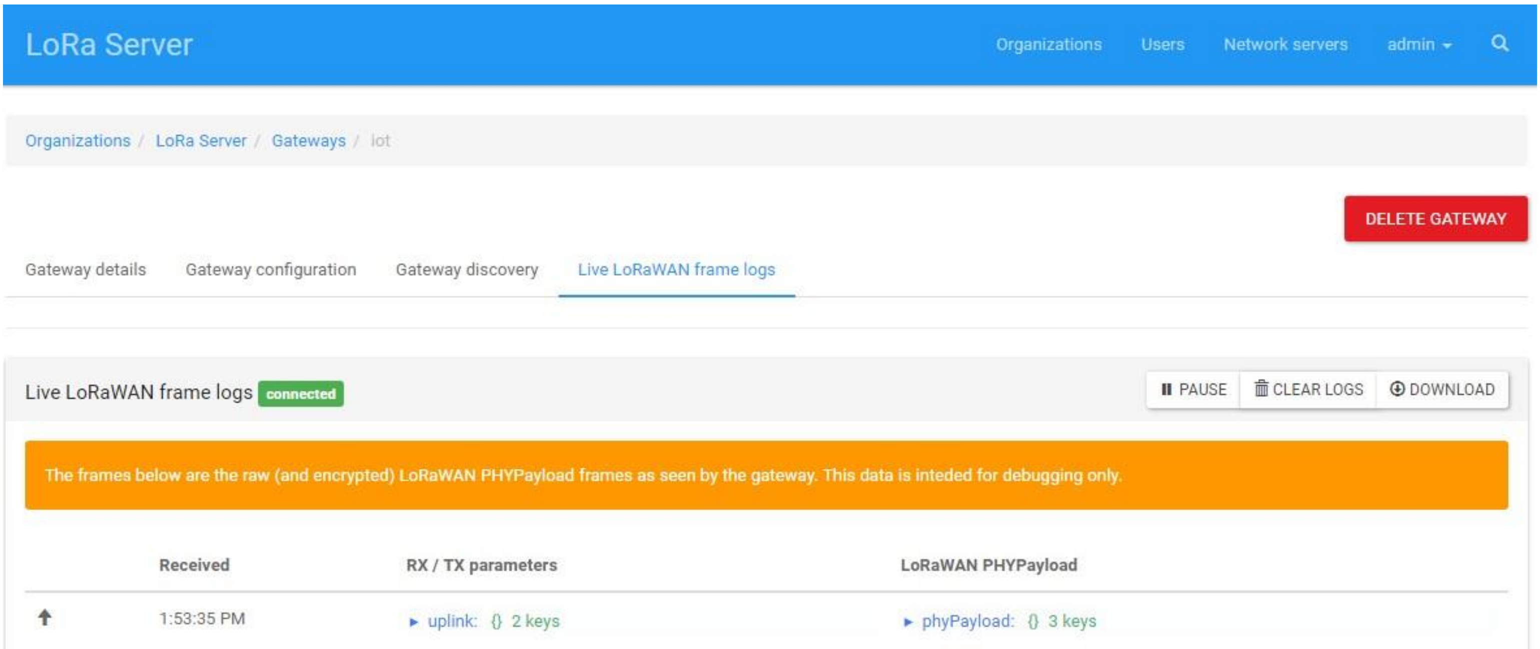
```
Sensing....
Value found: ,humidity=23.3,temperature=25.9Packet queued
Sensing....
Value found: ,humidity=23.3,temperature=25.9Packet queued
```

2. Encoded data will be then forwarded to the gateway bridge



```
root@raspberrypi:/etc/lora-app-server# mosquitto_sub -v -t "gateway/+/rx"

gateway/b827ebffff26ff5f/rx {"rxInfo":{"mac":"b827ebffff26ff5f","timestamp":1448439689,"frequency":868100000,"channel":0,"rfChain":0,"crcStatus":1,"codeRate":"4/5",
i":-55,"loRaSNR":9,"size":26,"dataRate":{"modulation":"LORA","spreadFactor":7,"bandwidth":125},"board":0,"antenna":0},"phyPayload":"QDYdASaA6AMBky13+pMUIszmgy1DAr8c
"}
```

3. In the application server's interface, we can visualize frames that are being received by the gateway

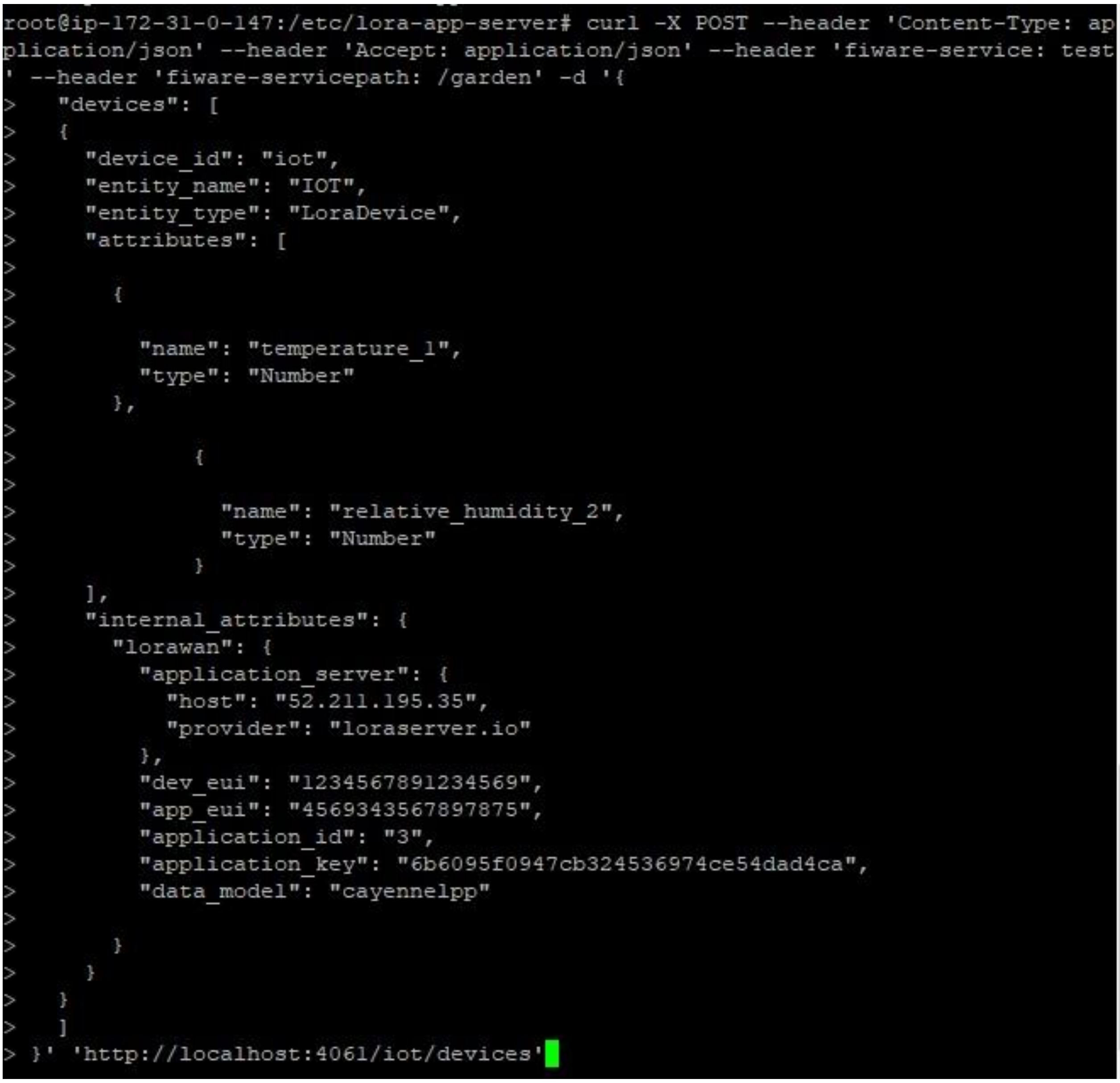


Live LoRaWAN frame logs connected

The frames below are the raw (and encrypted) LoRaWAN PHYPayload frames as seen by the gateway. This data is inteded for debugging only.

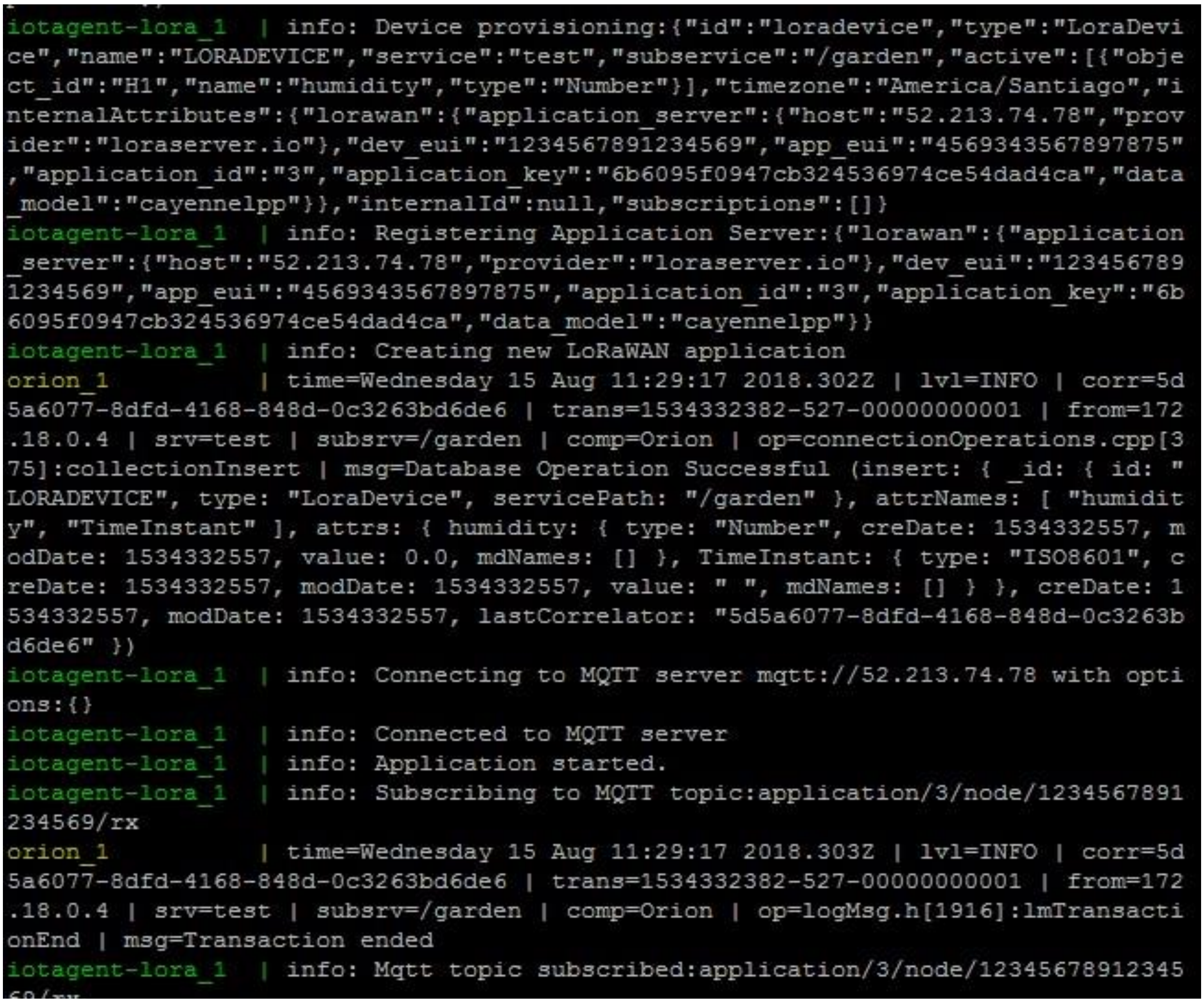
	Received	RX / TX parameters	LoRaWAN PHYPayload
↑	1:53:35 PM	▸ uplink: {} 2 keys	▸ phyPayload: {} 3 keys

4. Device provisioning: for the data to be forwarded to fiware's IoT agent through an MQTT broker, the device has to be provisioned and that is by: entering the attributes which, in our case, are 'temperature' and 'humidity' as well as providing loraserver's information, the device's eui, application eui, id , key and the data model which is Cayennelpp.



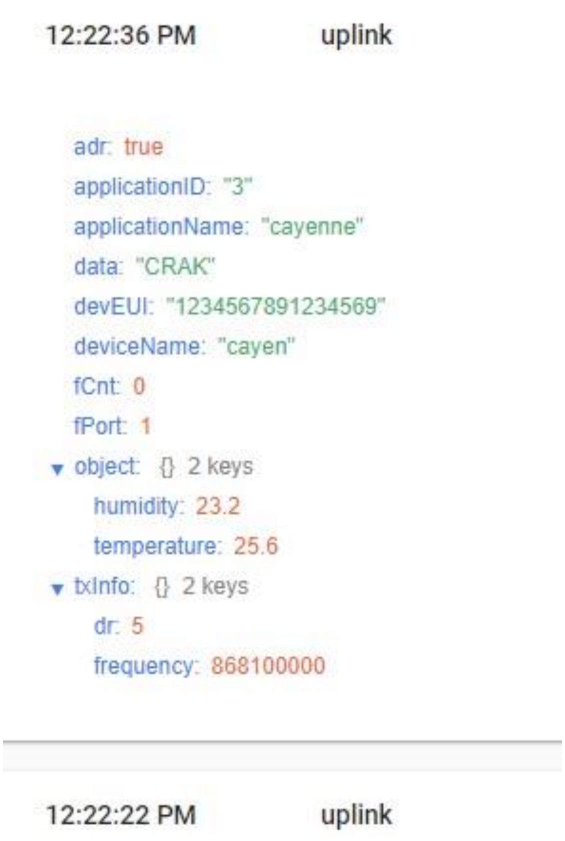
```
root@ip-172-31-0-147:/etc/lora-app-server# curl -X POST --header 'Content-Type: ap
plication/json' --header 'Accept: application/json' --header 'fiware-service: test
' --header 'fiware-servicepath: /garden' -d '{
>   "devices": [
>   {
>     "device_id": "iot",
>     "entity_name": "IOT",
>     "entity_type": "LoraDevice",
>     "attributes": [
>
>       {
>
>         "name": "temperature_1",
>         "type": "Number"
>       },
>
>             {
>
>               "name": "relative_humidity_2",
>               "type": "Number"
>             }
>     ],
>     "internal_attributes": {
>       "lorawan": {
>         "application_server": {
>           "host": "52.211.195.35",
>           "provider": "loraserver.io"
>         },
>         "dev_eui": "1234567891234569",
>         "app_eui": "4569343567897875",
>         "application_id": "3",
>         "application_key": "6b6095f0947cb324536974ce54dad4ca",
>         "data_model": "cayennelpp"
>
>       }
>     }
>   }
>   ]
> }' 'http://localhost:4061/iot/devices'
```

In the next figure, is the IoT agent processing the provisioning, connecting to the MQTT broker, starting the application and then subscribing to the application's topic



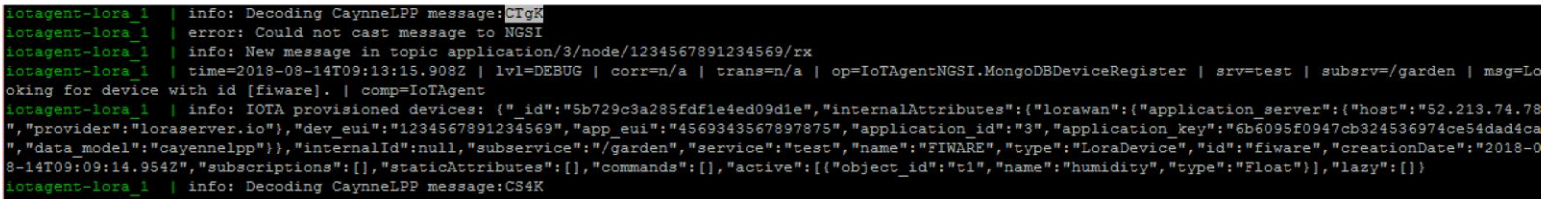
```
iotagent-lora_1  | info: Device provisioning:{"id":"loradevice","type":"LoraDevi
ce","name":"LORADEVICE","service":"test","subservice":"/garden","active":[{"obje
ct_id":"H1","name":"humidity","type":"Number"}],"timezone":"America/Santiago","i
nternalAttributes":{"lorawan":{"application_server":{"host":"52.213.74.78","prov
ider":"loraserver.io"},"dev_eui":"1234567891234569","app_eui":"4569343567897875"
,"application_id":"3","application_key":"6b6095f0947cb324536974ce54dad4ca","data
_model":"cayennelpp"}},"internalId":null,"subscriptions":[]}
iotagent-lora_1  | info: Registering Application Server:{"lorawan":{"application
_server":{"host":"52.213.74.78","provider":"loraserver.io"},"dev_eui":"123456789
1234569","app_eui":"4569343567897875","application_id":"3","application_key":"6b
6095f0947cb324536974ce54dad4ca","data_model":"cayennelpp"}}
iotagent-lora_1  | info: Creating new LoRaWAN application
orion_1          | time=Wednesday 15 Aug 11:29:17 2018.302Z | lvl=INFO | corr=5d
5a6077-8dfd-4168-848d-0c3263bd6de6 | trans=1534332382-527-00000000001 | from=172
.18.0.4 | srv=test | subsrv=/garden | comp=Orion | op=connectionOperations.cpp[3
75]:collectionInsert | msg=Database Operation Successful (insert: { _id: { id: "
LORADEVICE", type: "LoraDevice", servicePath: "/garden" }, attrNames: [ "humidit
y", "TimeInstant" ], attrs: { humidity: { type: "Number", creDate: 1534332557, m
odDate: 1534332557, value: 0.0, mdNames: [] }, TimeInstant: { type: "ISO8601", c
reDate: 1534332557, modDate: 1534332557, value: " ", mdNames: [] } }, creDate: 1
534332557, modDate: 1534332557, lastCorrelator: "5d5a6077-8dfd-4168-848d-0c3263b
d6de6" })
iotagent-lora_1  | info: Connecting to MQTT server mqtt://52.213.74.78 with opti
ons:{}
iotagent-lora_1  | info: Connected to MQTT server
iotagent-lora_1  | info: Application started.
iotagent-lora_1  | info: Subscribing to MQTT topic:application/3/node/1234567891
234569/rx
orion_1          | time=Wednesday 15 Aug 11:29:17 2018.303Z | lvl=INFO | corr=5d
5a6077-8dfd-4168-848d-0c3263bd6de6 | trans=1534332382-527-00000000001 | from=172
.18.0.4 | srv=test | subsrv=/garden | comp=Orion | op=logMsg.h[1916]:lmTransacti
onEnd | msg=Transaction ended
iotagent-lora_1  | info: Mqtt topic subscribed:application/3/node/12345678912345
69/rx
```

In the figure below, the received and decoded data in the application server is shown



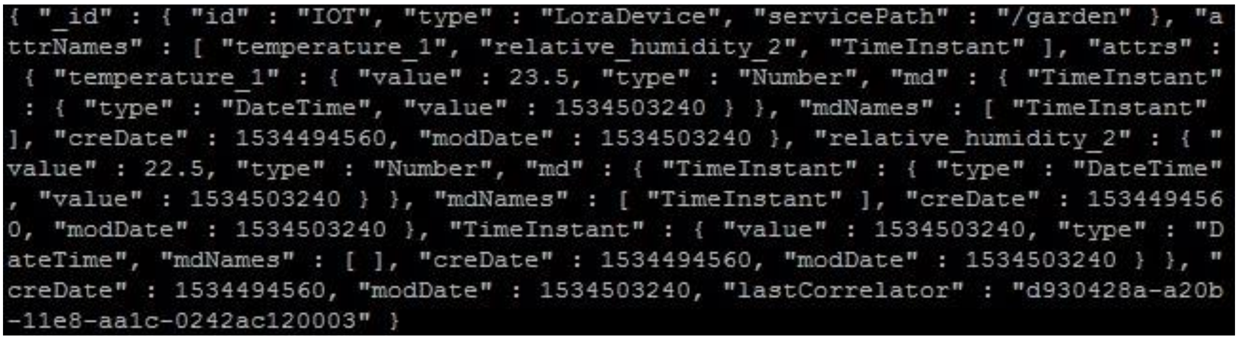
```
12:22:36 PM          uplink

  adr: true
  applicationID: "3"
  applicationName: "cayenne"
  data: "CRAK"
  devEUI: "1234567891234569"
  deviceName: "cayen"
  fCnt: 0
  fPort: 1
▼ object: {} 2 keys
    humidity: 23.2
    temperature: 25.6
▼ txInfo: {} 2 keys
    dr: 5
    frequency: 868100000

12:22:22 PM          uplink
```

In the following figure, data that has been forwarded to the IoT agent is shown



```
iotagent-lora_1  | info: Decoding CaynneLPP message:CTgK
iotagent-lora_1  | error: Could not cast message to NGSI
iotagent-lora_1  | info: New message in topic application/3/node/1234567891234569/rx
iotagent-lora_1  | time=2018-08-14T09:13:15.908Z | lvl=DEBUG | corr=n/a | trans=n/a | op=IoTAgentNGSI.MongoDBDeviceRegister | srv=test | subsrv=/garden | msg=Lo
oking for device with id [fiware]. | comp=IoTAgent
iotagent-lora_1  | info: IOTA provisioned devices: {"_id":"5b729c3a285fdf1e4ed09d1e","internalAttributes":{"lorawan":{"application_server":{"host":"52.213.74.78
","provider":"loraserver.io"},"dev_eui":"1234567891234569","app_eui":"4569343567897875","application_id":"3","application_key":"6b6095f0947cb324536974ce54dad4ca
","data_model":"cayennelpp"}},"internalId":null,"subservice":"/garden","service":"test","name":"FIWARE","type":"LoraDevice","id":"fiware","creationDate":"2018-0
8-14T09:09:14.954Z","subscriptions":[],"staticAttributes":[],"commands":[],"active":[{"object_id":"t1","name":"humidity","type":"Float"}],"lazy":[]}
iotagent-lora_1  | info: Decoding CaynneLPP message:CS4K
```

And below, is data stored in to context broker's database after being translated to NGSI by the IoT agent



```
{ "_id" : { "id" : "IOT", "type" : "LoraDevice", "servicePath" : "/garden" }, "a
ttrNames" : [ "temperature_1", "relative_humidity_2", "TimeInstant" ], "attrs" :
 { "temperature_1" : { "value" : 23.5, "type" : "Number", "md" : { "TimeInstant"
 : { "type" : "DateTime", "value" : 1534503240 } }, "mdNames" : [ "TimeInstant"
], "creDate" : 1534494560, "modDate" : 1534503240 }, "relative_humidity_2" : { "
value" : 22.5, "type" : "Number", "md" : { "TimeInstant" : { "type" : "DateTime"
, "value" : 1534503240 } }, "mdNames" : [ "TimeInstant" ], "creDate" : 153449456
0, "modDate" : 1534503240 }, "TimeInstant" : { "value" : 1534503240, "type" : "D
ateTime", "mdNames" : [ ], "creDate" : 1534494560, "modDate" : 1534503240 } }, "
creDate" : 1534494560, "modDate" : 1534503240, "lastCorrelator" : "d930428a-a20b
-11e8-aa1c-0242ac120003" }
```