HORIZON 2020 - PHOENIX

PLATFORM DEPLOYMENT AND GATEWAY CONFIGURATION GUIDE

Table of Contents

1. Introduction

This guide includes the information required to deploy and configure both the components of the PHOENIX platform that are running in the platform itself (core components, main agents, security components and other agents) as well as the base guidelines to be followed to configure the IoT Gateways.

In both cases the templates circulated in the project are used as reference by the designed scripts to simplify the provision task. For the IoT Gateways, once the relevant information of the devices has been identified, the final configuration files can be directly generated.

As for the platform, once the template is ready, all the entities and subscriptions are automatically created and each component's configuration is populated accordingly.

2. Platform configuration

2.1. Requirements

The software is installed using Docker.

To do that, a Virtual Machine is used with the following requirements:

- Operative system: Ubuntu, Debian or CentOS.
- Docker Engine: A quick installation guide for Ubuntu can be found here https://docs.docker.com/engine/install/ubuntu/
- Docker Compose: The installation instructions can be found here https://docs.docker.com/compose/install/

2.2. Core components

The software includes the following components:

- **eclipse-mosquitto**
- **mongo**
- **fiware/orion-ld**
- **fiware/iotagent-json**
- **phoenix/storage-component**

Here we can see the contents of the **docker-compose.yml** file:

```
version: "3.5"

services:
```

```
  mosquitto:
    image: eclipse-mosquitto
    container_name: mosquitto
    hostname: mosquitto
    restart: always
    ports:
      - "1883:1883"
      - "8883:8883"
    environment:
      - TZ=Europe/Madrid
    volumes:
      - "./mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf"
      - "./mosquitto/certs:/mosquitto/config/certs"
      - "./mosquitto/conf.d:/mosquitto/config/conf.d"
      - "./mosquitto/aclfile:/mosquitto/config/aclfile"
      - "./mosquitto/aclfileSSL:/mosquitto/config/aclfileSSL"
      - "./mosquitto/passwd:/mosquitto/config/passwd"
      - "./mosquitto/data:/mosquitto/data"
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"

  mongo-db:
    image: mongo:4.2
    hostname: mongo-db
    container_name: db-mongo
    restart: always
    expose:
     - "27017"
    ports:
     - "127.0.0.1:27017:27017"
    volumes:
     - mongo_data:/data/db
    healthcheck:
     test: |
       host=`hostname --ip-address || echo '127.0.0.1'`;
       mongo --quiet $host/test --eval 'quit(db.runCommand({ ping: 1 }).ok ? 0 : 2)' &&
echo 0 || echo 1
     interval: 5s
    logging:
     driver: "json-file"
     options:
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
      max-size: "10m"
      max-file: "3"

orion:
  image: fiware/orion-ld
  hostname: orion
  container_name: fiware-orion
  depends_on:
    - mongo-db
  restart: always
  ports:
    - "1026:1026"
  expose:
    - "1026"
  command: -dbhost mongo-db -logLevel DEBUG -forwarding -corsOrigin __ALL
  healthcheck:
    test: curl --fail -s http://orion:1026/version || exit 1
    interval: 5s
  logging:
    driver: "json-file"
    options:
      max-size: "10m"
      max-file: "3"

iot-agent:
  image: fiware/iotagent-json
  hostname: iot-agent
  container_name: fiware-iot-agent
  depends_on:
    - mongo-db
  restart: always
  expose:
    - "4041"
  ports:
    - "4041:4041"
  environment:
    - IOTA_CB_URL=http://orion:1026
    - IOTA_CB_HOST=orion
    - IOTA_CB_PORT=1026
    - IOTA_NORTH_PORT=4041
    - IOTA_LOG_LEVEL=DEBUG
    - IOTA_TIMESTAMP=true
    - IOTA_CONFIG_RETRIEVAL=false
    - IOTA_DEFAULT_KEY=PHOENIX-ODINS
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
    - IOTA_DEFAULT_TRANSPORT=MQTT
    - IOTA_AUTOCAST=true
    - IOTA_REGISTRY_TYPE=mongodb
    - IOTA_MONGO_HOST=mongo-db
    - IOTA_MONGO_PORT=27017
    - IOTA_MONGO_DB=iotagentjson
    - IOTA_HTTP_PORT=7896
    - IOTA_PROVIDER_URL=http://iot-agent:4041
    - IOTA_DEVICE_REGISTRATION_DURATION=P40Y
    - IOTA_DEFAULT_RESOURCE=/iot/json
    - IOTA_CB_NGSI_VERSION=ld
    -               IOTA_JSON_LD_CONTEXT=https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-
context.jsonld
    - IOTA_FALLBACK_TENANT=phoenix
    - IOTA_MQTT_PROTOCOL=mqtt
    - IOTA_MQTT_HOST=phoenix.odins.es
    - IOTA_MQTT_PORT=1883
    - IOTA_MQTT_USERNAME=GRVauKErF7Tu1uAeaur9kcNriiGi3I/f
    - IOTA_MQTT_PASSWORD=ZTUFBwIo8sY/tHI1JoPoN8nvTFNG+qJr
    - IOTA_MQTT_QOS=0
    - IOTA_MQTT_RETAIN=false
    - IOTA_MQTT_RETRIES=5
    - IOTA_MQTT_RETRY_TIME=5
    - IOTA_MQTT_KEEPALIVE=60
    - IOTA_MQTT_AVOID_LEADING_SLASH=false
    - IOTA_SERVICE=phoenix
    - IOTA_SUBSERVICE=/phoenix
  healthcheck:
    interval: 5s
  logging:
    driver: "json-file"
    options:
      max-size: "10m"
      max-file: "3"

 storage-component:
  image: phoenix/storage-component
  hostname: storage-component
  container_name: storage-component
  restart: always
  volumes:
    - storage_component_data:/var/lib/storage-data
  ports:
    - "8666:8666"
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"

volumes:
  mongo_data:
    name: mongo_data
    external: true
  storage_component_data:
    name: storage_component_data
    external: true
```

2.3. Security components

The software includes the following components:

- **fiware/idm:7.8.1** for Keyrock, the FIWARE Identity Manager.
- **mysql:5.7** used by Keyrock.
- **phoenix/capability-manager** for the Capability Manager.
- **phoenix/pap-pdp** for the XACML Access Control Framework.
- **phoenix/pep-proxy** as the proxy element of the Security & Privacy Framework (multiple instances deployed)

Here we can see the contents of the **docker-compose.yml** file:

```
version: "3.5"
services:
  # Keyrock is an Identity Management Front-End
  # This container is used as an authentication system and it also takes part in authorisation
  keyrock:
    build: keyrock/. # Modified image of Keyrock to add certificates
    image: phoenix/idm:7.8.1
    container_name: fiware-keyrock
    hostname: keyrock
    restart: always
    networks:
      security_network:
        ipv4_address: 172.20.0.6
    ports:
      - "5443:3443" # runs KEyrock in HTTPS protocol
    depends_on:
      - mysql-db
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
  volumes:
    -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.o
dins.es/fullchain.pem:/opt/fiware-idm/certs/idm-2018-cert.pem    # volumes to
import certificates to container
    -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.o
dins.es/privkey.pem:/opt/fiware-idm/certs/idm-2018-key.pem

  environment:
    - DEBUG=idm:*
    - IDM_TITLE=Access Control System
    - IDM_DB_HOST=mysql-db
    - IDM_DB_PASS_FILE=/run/secrets/my_secret_data # file with database password
in same directory as this file
    - IDM_DB_USER=root # mysql keyrock database user
    - IDM_HOST=http://localhost:3005 # IP address or domain where Keyrock is
running (host machine IP or domain)
    - IDM_PORT=3005 # port where Keyrock runs in HTTP
    - IDM_HTTPS_ENABLED=true # enable or diable (true / false) HTTPS protocol in
Keyrock
    - IDM_HTTPS_PORT=3443 # port where Keyrock runs HTTPS in case of being
enabled
    - IDM_ADMIN_USER=admin # Keyrock administrator username
    - IDM_ADMIN_EMAIL=admin_phoenix@test.com # Keyrock administrator mail
    - IDM_ADMIN_PASS=phoenix # Keyrock administrator password
    - IDM_PDP_LEVEL=advanced # Authorisation level (basic - HTTP rules enough with
Keyrock or advanced - AXCML rules with AuthZforce and Wilma PEP Proxy)
    #- IDM_AUTHZFORCE_ENABLED=false # Enable or disable AuthZforce to use
advanced PDP authorisation level with AXCML rules
    - IDM_OAUTH_EMPTY_STATE=true
  secrets:
    - my_secret_data # File where mysql password is hidden
  healthcheck:
    test: curl --fail -s http://localhost:3005/version || exit 1 # assures correct
behaviour

 # Database (Keyrock)
 mysql-db:
  restart: always # In case it takes too long to load database
  image: mysql:5.7
  hostname: mysql-db
  container_name: db-mysql
  expose:
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
    - "3306" # service runs on this port, enable communication between database and
Keyrock
   ports:
    - "3307:3306"
   networks:
    security_network:
     ipv4_address: 172.20.0.11
   environment:
    - "MYSQL_ROOT_PASSWORD_FILE=/run/secrets/my_secret_data" # password for
root administrator of database
    - "MYSQL_ROOT_HOST=172.20.0.6" # Allow Keyrock to access this database
   volumes:
    - mysql-db:/var/lib/mysql
   secrets:
    - my_secret_data # file where root password is hidden

 ######
 ### CapabilityManager
 ######
 capabilitymanager:
   build: Py_CapabilityManagerWebService/.
   image: phoenix/capability-manager
   logging:
    driver: "json-file"
    options:
     max-size: "10m"
     max-file: "3"
   ports:
     - "3030:3030"
   networks:
    security_network:
     ipv4_address: 172.20.0.2

   restart: always
   volumes:
     -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.o
dins.es/fullchain.pem:/opt/API-CM/certs/server-public-cert.pem    # volumes to
import certificates to container
     -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.o
dins.es/privkey.pem:/opt/API-CM/certs/server-priv-rsa.pem
     - ./capabilitymanager.log:/opt/API-CM/out.log
   environment:
```

```
      #- capman_protocol=http #(OPTIONAL default https)
      - keyrock_protocol=https
      - keyrock_host=172.20.0.6 #<specify IdM Public IP address>
      - keyrock_port=3443
      - keyrock_admin_email=admin_phoenix@test.com
      - keyrock_admin_pass=phoenix

      #Validate Capability token using blockchain: Admitted values: "0: No use; 1:Use"
      - blockchain_usevalidation=0

      #BlockChain protocol. Admitted values: "http","https"
      - blockchain_protocol=http
      #BlockChain host.
      - blockchain_host=localhost #<specify BlockChain Public IP address>
      - blockchain_port=8000

      #PDP protocol. Admitted values: "http"
      - PDP_protocol=http
      #PDP host.
      - PDP_host=172.20.0.3
      - PDP_port=8080

######
### XACML
######
xacml:
  build: XACML_PAP_PDP/.
  image: phoenix/pap-pdp
  logging:
    driver: "json-file"
    options:
      max-size: "10m"
      max-file: "3"
  expose:
    - "8080"
  ports:
    - "8080:8080"
  networks:
    security_network:
      ipv4_address: 172.20.0.3

  restart: always

  # ALL environment variables are optionals.
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
  # BlockChain_integration to indicate if blockchain integration is considered or not.
  # The rest of BlockChain_* variables are used if blockchain integration is considered.
  environment:
    #BlockChain_integration : admittable values:
    # 0-No integration with blockchain
    # 1-Integration with blockchain
    # Default value : 0
    - BlockChain_integration=0

    #BlockChain_configuration : admittable values:
    # 0-Uses configuration from blockchain.conf file.
    # 1-Uses configuration from folling environment variables.
    - BlockChain_configuration=0 # Optional: Default value : 0

    #Environment variables are considered only if BlockChain_integration=1
    - BlockChain_protocol=http # Optional: Default value : http
    - BlockChain_domain=domaintest #<specify Blockchain dommain> # Required
    - BlockChain_IP=localhost #<specify Blockchain endpoint IP address> # Required
    - BlockChain_port=8000 # Optional Default value : 8000

    - BlockChain_get_resource=/policy/domaintest # Optional : Default value :
/policy/domaintest
    - BlockChain_post_resource=/policy/register # Optional : Default value :
/policy/register
    - BlockChain_update_resource=/policy/update # Optional : Default value :
/policy/update

  volumes:
    - ./continue-a.xml:/usr/local/tomcat/PAPConfigData/Policies/continue-a.xml
    -
./XACML_Attributes.xml:/usr/local/tomcat/PAPConfigData/XACMLAtts/XACML_Attri
butes.xml

 ######
 ### pepproxy
 ######
 pepproxy:
  build: Py_PEP-Proxy/.
  image: phoenix/pep-proxy
  logging:
   driver: "json-file"
   options:
    max-size: "10m"
    max-file: "3"
```

```
    ports:
      - "1030:1027"
    networks:
     security_network:
      ipv4_address: 172.20.0.5

    restart: always
    volumes:
      -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/fullchain.pem:/opt/PEP-Proxy/certs/server-public-cert.pem  # volumes to import certificates to container
      -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/privkey.pem:/opt/PEP-Proxy/certs/server-priv-rsa.pem
      - ./pepproxy.log:/opt/PEP-Proxy/out.log
    environment:
      #- pep_protocol=http #(OPTIONAL default https)
      #Broker protocol. Admitted values: "http","https"
      - target_protocol=http
      #Broker host.
      - target_host=phoenix.odins.es #Target Production endpoint
      - target_port=1026
      #Broker API. Admitted values: "NGSIv1","NGSIv2","NGSILDv1","GenericAPI"
      - target_API=NGSILDv1 #<specify Broker API type>#

      #Validate Capability token using blockchain: Admitted values: "0: No use; 1:Use"
      - blockchain_usevalidation=0

      #BlockChain protocol. Admitted values: "http","https"
      - blockchain_protocol=http
      #BlockChain host.
      - blockchain_host=localhost #<specify BlockChain Public IP address>
      - blockchain_port=8000

      # PEP proxy endpoint: protocol+ip+port
      # HOST NO admitted: 0.0.0.0, localhost, 127.0.0.1
      -  PEP_ENDPOINT=https://phoenix.odins.es:1030  #<specify PEP-Proxy Public address ex: https://<PEP-IP>:<PEP-PORT>>

  ######
  ### pepproxyzwave
  ######
```


Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

```
  pepproxyzwave:
    build: Py_PEP-Proxy/.
    image: phoenix/pep-proxy
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"
    ports:
      - "1028:1027"
    networks:
      security_network:
        ipv4_address: 172.20.0.7

    restart: always
    volumes:
      -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/fullchain.pem:/opt/PEP-Proxy/certs/server-public-cert.pem  # volumes to import certificates to container
      -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/privkey.pem:/opt/PEP-Proxy/certs/server-priv-rsa.pem
      - ./pepproxy.log:/opt/PEP-Proxy/out.log
    environment:
      #- pep_protocol=http #(OPTIONAL default https)
      #Broker protocol. Admitted values: "http","https"
      - target_protocol=http
      #Broker host.
      - target_host=phoenix.odins.es #Target Production endpoint
      - target_port=1880
      #Broker API. Admitted values: "NGSIv1","NGSIv2","NGSILDv1","GenericAPI"
      - target_API=GenericAPI #<specify Broker API type>#

      #Validate Capability token using blockchain: Admitted values: "0: No use; 1:Use"
      - blockchain_usevalidation=0

      #BlockChain protocol. Admitted values: "http","https"
      - blockchain_protocol=http
      #BlockChain host.
      - blockchain_host=localhost #<specify BlockChain Public IP address>
      - blockchain_port=8000

      # PEP proxy endpoint: protocol+ip+port
```

```
      # HOST NO admitted: 0.0.0.0, localhost, 127.0.0.1
      - PEP_ENDPOINT=https://phoenix.odins.es:1028 #<specify PEP-Proxy Public address ex: https://<PEP-IP>:<PEP-PORT>>

 ######
 ### pepproxystorage
 ######
 pepproxystorage:
  build: Py_PEP-Proxy/.
  image: phoenix/pep-proxy
  logging:
   driver: "json-file"
   options:
    max-size: "10m"
    max-file: "3"
  ports:
    - "1029:1027"
  networks:
   security_network:
    ipv4_address: 172.20.0.8

  restart: always
  volumes:
    -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/fullchain.pem:/opt/PEP-Proxy/certs/server-public-cert.pem  # volumes to import certificates to container
    -
/home/phoenix/deployment/letsencrypt_certbot/data/certbot/conf/live/phoenix.odins.es/privkey.pem:/opt/PEP-Proxy/certs/server-priv-rsa.pem
    - ./pepproxystorage.log:/opt/PEP-Proxy/out.log
  environment:
    #- pep_protocol=http #(OPTIONAL default https)
    #Broker protocol. Admitted values: "http","https"
    - target_protocol=http
    #Broker host.
    - target_host=phoenix.odins.es #Target Production endpoint
    - target_port=8666
    #Broker API. Admitted values: "NGSIv1","NGSIv2","NGSILDv1","GenericAPI"
    - target_API=GenericAPI #<specify Broker API type>#

    #Validate Capability token using blockchain: Admitted values: "0: No use; 1:Use"
    - blockchain_usevalidation=0
```

```
      #BlockChain protocol. Admitted values: "http","https"
      - blockchain_protocol=http
      #BlockChain host.
      - blockchain_host=localhost #<specify BlockChain Public IP address>
      - blockchain_port=8000

      # PEP proxy endpoint: protocol+ip+port
      # HOST NO admitted: 0.0.0.0, localhost, 127.0.0.1
      - PEP_ENDPOINT=https://phoenix.odins.es:1029 #<specify PEP-Proxy Public
address ex: https://<PEP-IP>:<PEP-PORT>>

networks:
  security_network:
    driver_opts:
      com.docker.network.bridge.name: br_security
    ipam:
      config:
        - subnet: 172.20.0.0/16

volumes:
  mysql-db: ~

secrets:
  my_secret_data:
    file: ./keyrock/secrets.txt # Path for hidden-password file
```

From a configuration point of view, the first step to follow is to identify the location of the components that are going to be protected (host, port, protocol, endpoint, etc.).

The most relevant variables are those related to the instances of the PEP Proxy, namely:

- **target_protocol**: Protocol used for communicating with the internal component (either http or https).
- **target_host**: Host where the internal component is listening.
- **target_port**: Listen port of the internal component.
- **target_API**: API used by the internal component (required for cyphering support) (NGSIv1, NGSIv2, NGSILDv1 or GenericAPI).
- **PEP_ENDPOINT**: URL of the resource.

2.4. Other agents/integrations

Most of the other agents of the platform run on top of Node-RED.

Here we can see the contents of the **docker-compose.yml** file:

```
version: "3.5"

services:
  nodered:
    image: nodered/node-red:latest
    restart: always
    ports:
      - 1880:1880
    volumes:
      - "nodered_data:/data"
      - "./settings.js:/data/settings.js"
      - "./configZ-WaveActuationAgent.json:/data/configZ-WaveActuationAgent.json"
      - "./statusIntegrationENTSOE:/data/statusIntegrationENTSOE"
      - "./statusIntegrationREE:/data/statusIntegrationREE"
      - "./statusIntegrationUMUScada:/data/statusIntegrationUMUScada"
    environment:
      - TZ=Europe/Madrid

volumes:
  nodered_data:
```

2.4.1. ENTSO-E

Regarding the ENTSO-E integration, the following figures show how a new country can be added to the configuration (through the Node-RED web interface):

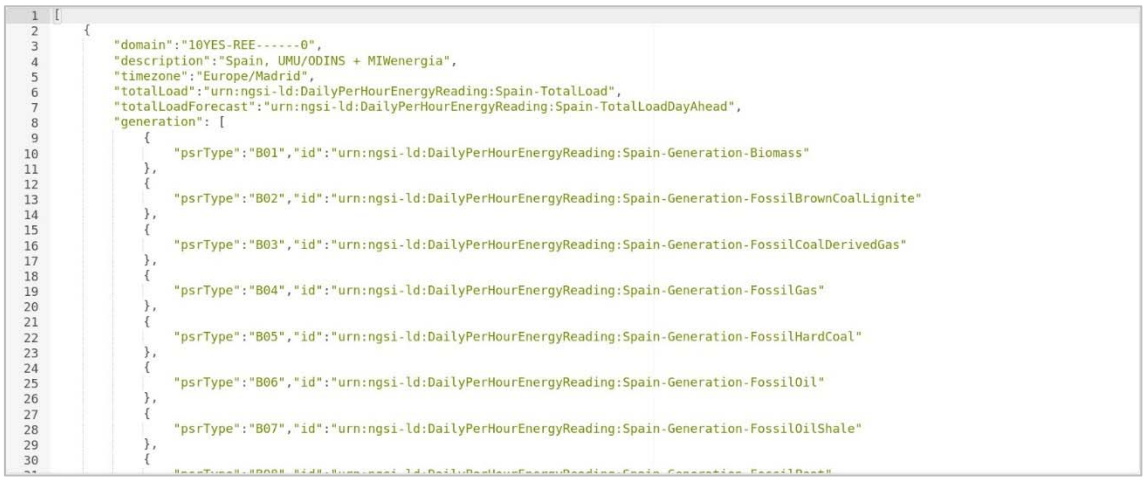

Figure 1: Configuration of entities and other API-dependent information

Odin Solutions, S.L | Calle Palma de Mallorca 2 | 30009 – Murcia | +34 902 570 121 | info@odins.es

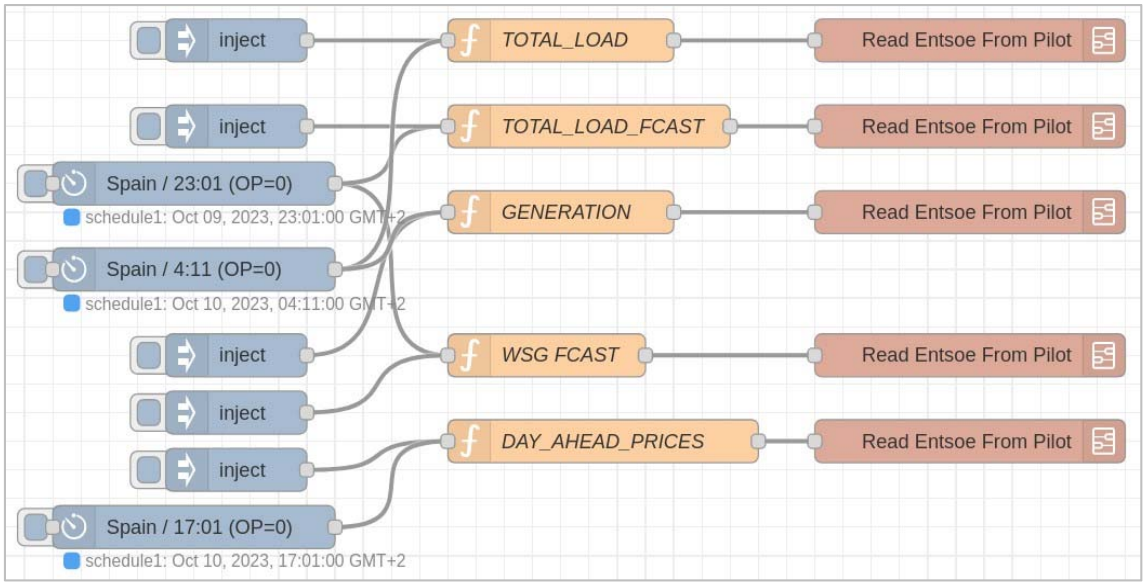

Figure 2: Configuration of the scheduling for each type of reading

3. Gateway configuration

3.1. Requirements

The IoT Gateways used in the project are *Raspberry Pi*, which use an image developed in PHOENIX. This image is based on *Rasberry OS* (formerly *Raspbian*), a distribution based on *Debian*.

Depending on the technology used by the devices to be controlled, additional hardware could be required:

- Modbus → USB ↔ RS-485 adapter.
- Z-Wave → USB Z-Wave stick (i.e. *Aeotec Z-Stick Gen5* for *Raspberry Pi 3b+* or *Aeotec Z-Stick Gen5+* for *Raspberry Pi 4*).

3.2. Modbus and Modbus/TCP

Regardless of the physical interface (RS-485, Ethernet or WiFi), the designed scripts directly generate the configuration files. Since this is fully dependent on the manufacturer, there is no automatic way to do that, and that is why scripts for well-known devices were created (for those used in the different pilots).

3.3. Z-Wave

In this case the configuration is obtained through the *Z-Wave JS Control Panel* (included in the *Z-Wave JS* plugin of Node-RED, also installed in the image).

Once the devices have been paired with the USB stick, they are automatically detected by the *Z-Wave JS* library as can be seen in the following figure:

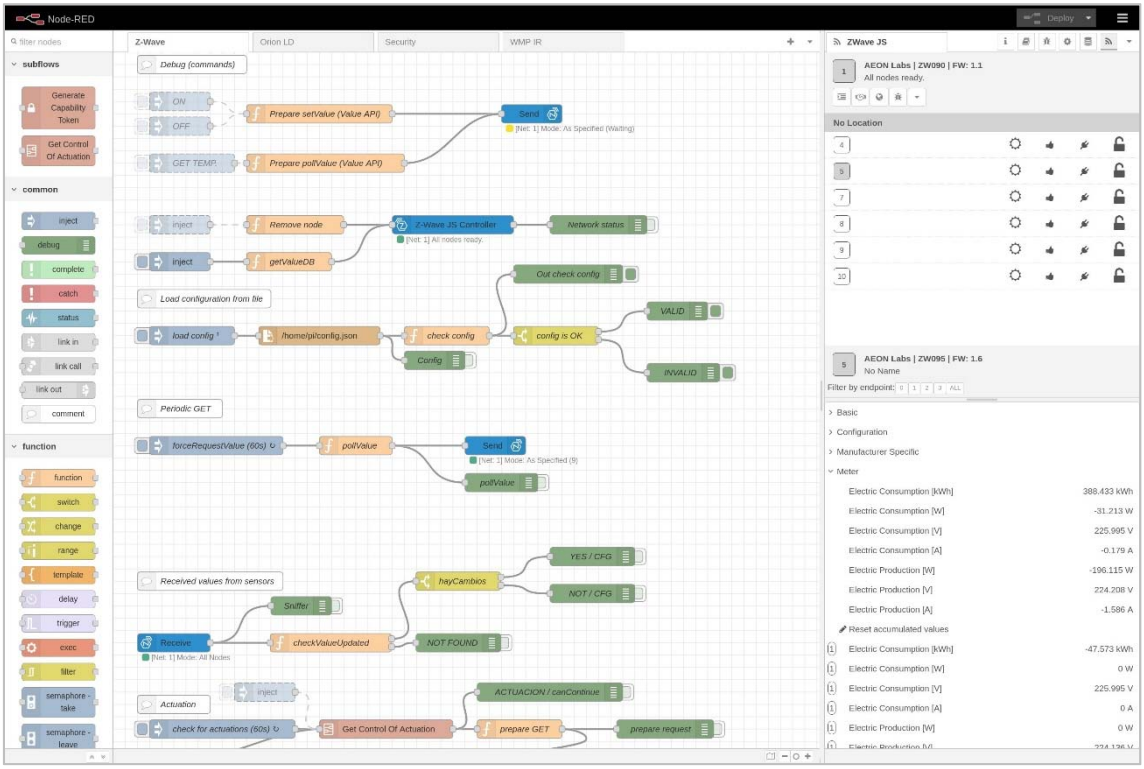

Figure 3: Z-Wave JS Control Panel (right)

Figure 4: Configuration of a single reading (valueId) (double click on individual entries in the bottom-right panel, inside each Command Class)