Plug-N-Harvest

WP3: THE PLUG-N-HARVEST MONITORING RELEASE AND GATEWAY MANAGEMENT

ORGANIZATION: CERTH, ODINS
PRESENTER(S): IAKOVOS MICHAILIDIS, STELIOS KRINIDIS, DAN GARCIA, RAFAEL MARIN-PEREZ
MEETING: 7TH PLENARY MEETING 26-27 FEBRUARY 2020, BRUSSELS, BELGIUM

Plug-N-Harvest: Project Information

Cordis Europa URL: http://cordis.europa.eu/project/rcn/211287_en.html

Project Website: www.plug-n-harvest.eu

Project Acronym : PLUG-N-HARVEST

Project ID: 768735

Funded under: H2020-EU.2.1.5.2. - Technologies enabling energy-efficient systems and energy-efficient buildings with a low environmental impact

Project Start Date: 1st of September 2017

Duration: 51 months

List of Participants	
1	Centre for Research and Technology Hellas - CERTH
2	Rheinisch-Westfaelische Technische Hochschule Aachen - RWTH
3	Cardiff University – CU
4	Aloumyl, Biomichania Alouminioy Anonimi Etairia - ALUMIL
5	Sistemes Avancats De Energia Solar Termica Sccl - AIGUASOL
6	Odin Solutions s.l. - ODINS
7	SIEMENS SRL - SIE
8	Etra Investigacion Y Desarrollo Sa - ETRA
9	Energy Transitions Limited - ET
10	Eco Intelligent Growth, SL - EIG
11	Agencia De L'habitatge De Catalunya - AHC
12	Perifieria Dytikhs Makedonias - RWM
13	County Council Of The City And County Of Cardiff - CCC

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

	Mongo Database Manager (OdinS)
Hosting platform/environment	Mongo DB, MySQL, PhP
Mission/Goal	Collection, storage and retrieving historical data
Methodology considered	Based on a standardized COMET component integrated in the royalty-free FIWARE platform.
Input Data (and its origin)	Real-time sensor data from IoT gateways, weather retrieving module and comfort calculating module.
Output Data (and its purpose)	Historical data for showing in graphical user interfaces and for processing in energy management systems.

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

	Data Manager for PC and RasPi gateways (CERTH)
Hosting platform/environment	Python
Mission/Goal	Collecting data from sensors/actuators; Push measurement data to the BMS server; Push data control decisions from the BMS server to the actuators;
Methodology considered	Datetime; Configparser; Timedelta; Sys
Input Data (and its origin)	Bi-directional communication: Real-time sensor data from sensor/actuator APIs; Real-time control data from BMS (IMCS decisions);
Output Data (and its purpose)	Bi-directional communication: Real-time sensor/actuator data to BMS server; Real-time IMCS control data to actuator APIs;

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

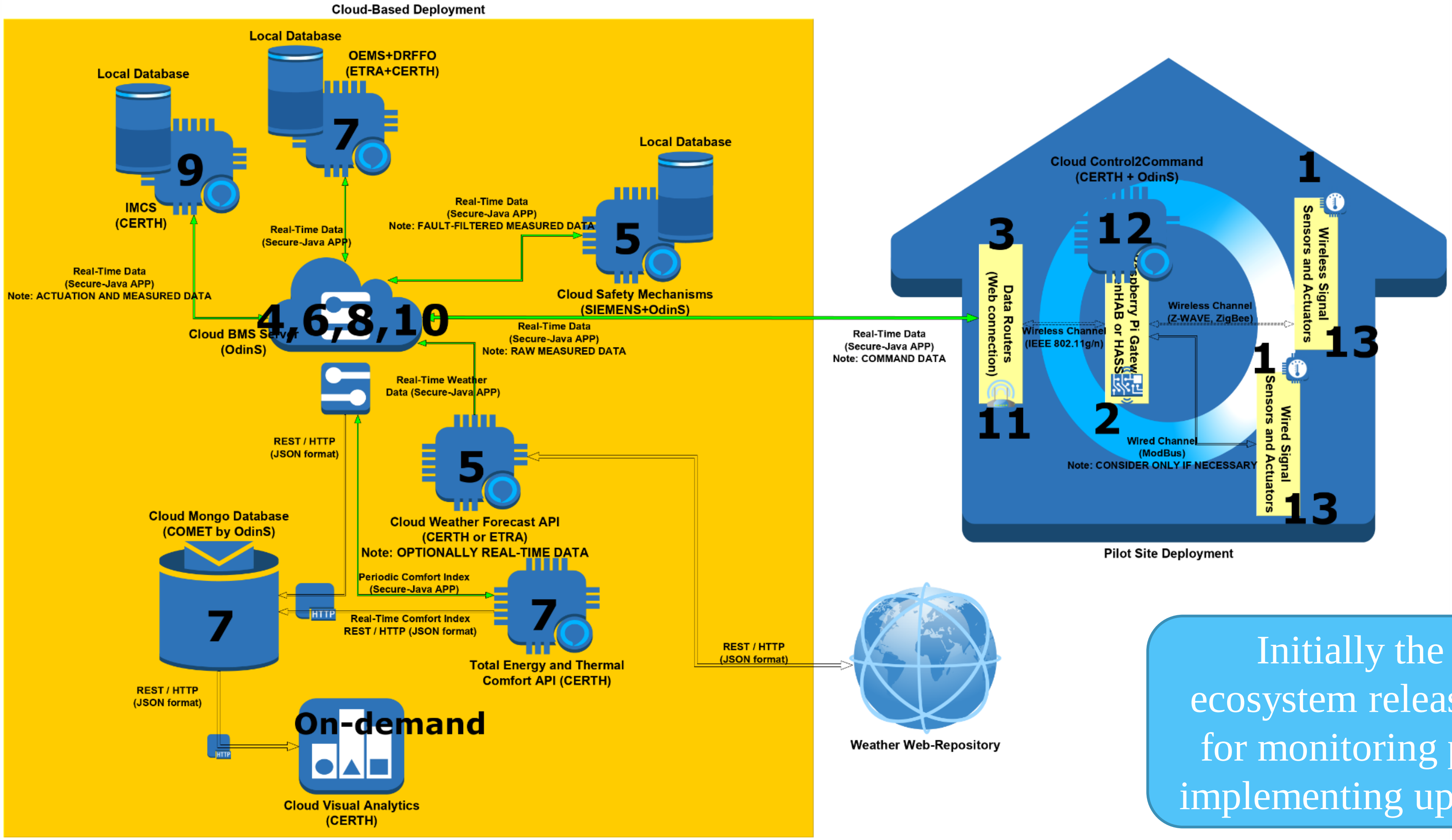
	Weather Forecast Retrieving (ETRA)
Hosting platform/environment	Daemon located on a web-server which periodically asks to OpenWeatherMap or tutiempo for weather information
Mission/Goal	Collect weather data for each building's weather conditions and forecasts.
Methodology considered	Backend process calling OpenWeatherMap or tutiempo
Input Data (and its origin)	Weather data received from OpenWeatherMap or tutiempo service
Output Data (and its purpose)	Current weather conditions and weather forecasts for each building sent and saved in common database.

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

	Graphical User Interface (CERTH)
Hosting platform/environment	Plug-N-Harvest GUI is angular JS based, and it will be located in CERTH smart home (Linux environment).
Mission/Goal	Goal of the GUI is to visualize the building and district information to the end-users.
Methodology considered	Visual analytics for intuitive user awareness
Input Data (and its origin)	The buildings' information provided by the multi-sensorial network, as well as by the plug-N-Harvest modules.
Output Data (and its purpose)	Visualization widgets.

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

	Total Energy and Thermal Comfort Calculating
Hosting platform/environment	Server located in CERTH – Greece, running in Linux platform, operating in Python environment.
Mission/Goal	Calculate occupants comfort with respect to building's conditions.
Methodology considered	Python script operating as a sub-module of the IMCS module, utilizing Fanger's equations.
Input Data (and its origin)	Each building's internal conditions (coming from BMS server) and current weather conditions (coming from BMS server and weather module).
Output Data (and its purpose)	Percentage of Dissatisfied People in each room.



Initially the WP3 ecosystem release will be for monitoring purposes implementing up to **Step 8**

Plug-N-Harvest: WP3 Pilot-Monitoring Modules

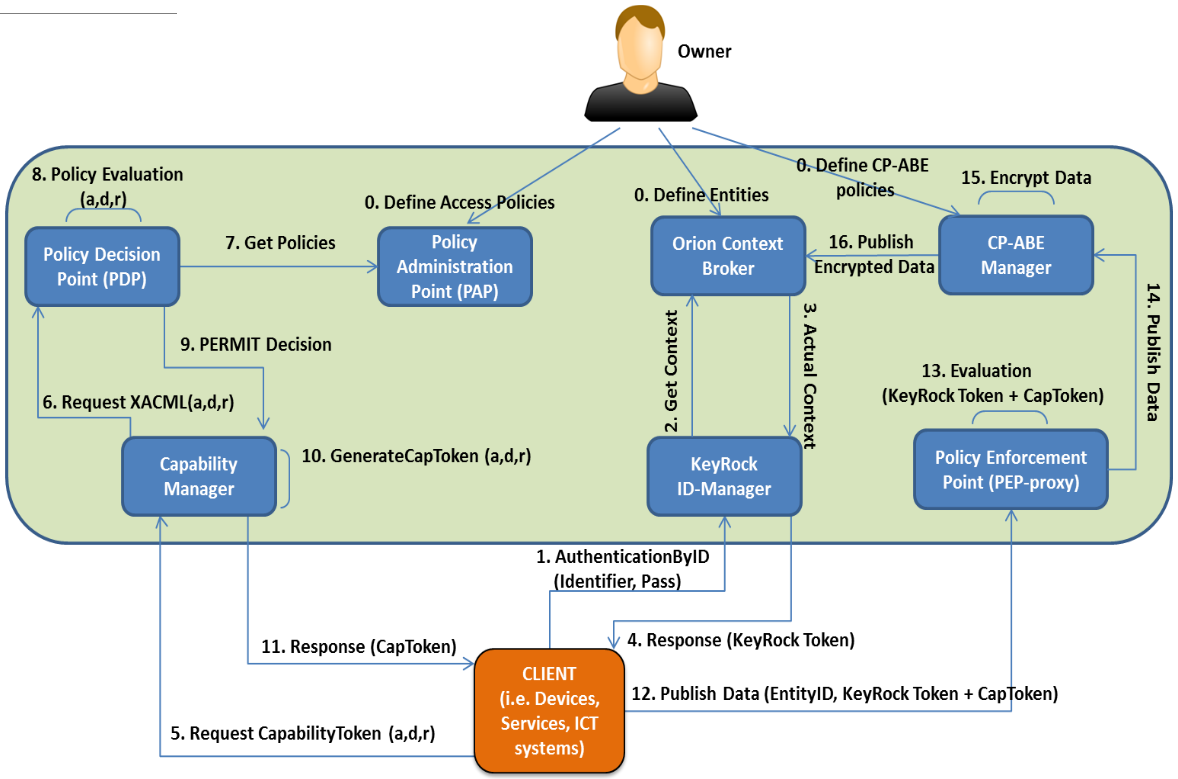
Execution Sequence:

The integrated fully functional cloud ecosystem implements a control loop (T=15mins) as depicted in the figure above.

Plug-N-Harvest: BMS Communication & Execution Management [OdinS]

Plug-N-Harvest: BMS Server with Security/Privacy Functions

- ❑The BMS is based on open FIWARE platform and is ready for the pilots testing:
  - ❑Context Broker for scalable management of real-time data about sensing/actuation operations.
  - ❑IoT Agents to enable high connectivity of heterogeneous entities (i.e. devices and ICT systems).
  - ❑COMET (Historical data without secure and privacy policies)


Owner
8. Policy Evaluation (a,d,r)
0. Define Access Policies
0. Define Entities
0. Define CP-ABE policies
15. Encrypt Data
Policy Decision Point (PDP)
7. Get Policies
Policy Administration Point (PAP)
Orion Context Broker
16. Publish Encrypted Data
CP-ABE Manager
14. Publish Data
9. PERMIT Decision
2. Get Context
3. Actual Context
13. Evaluation (KeyRock Token + CapToken)
6. Request XACML(a,d,r)
Capability Manager
10. GenerateCapToken (a,d,r)
KeyRock ID-Manager
Policy Enforcement Point (PEP-proxy)
1. AuthenticationByID (Identifier, Pass)
11. Response (CapToken)
4. Response (KeyRock Token)
CLIENT (i.e. Devices, Services, ICT systems)
12. Publish Data (EntityID, KeyRock Token + CapToken)
5. Request CapabilityToken (a,d,r)

Plug-N-Harvest: Secure Components for Data exchange with BMS

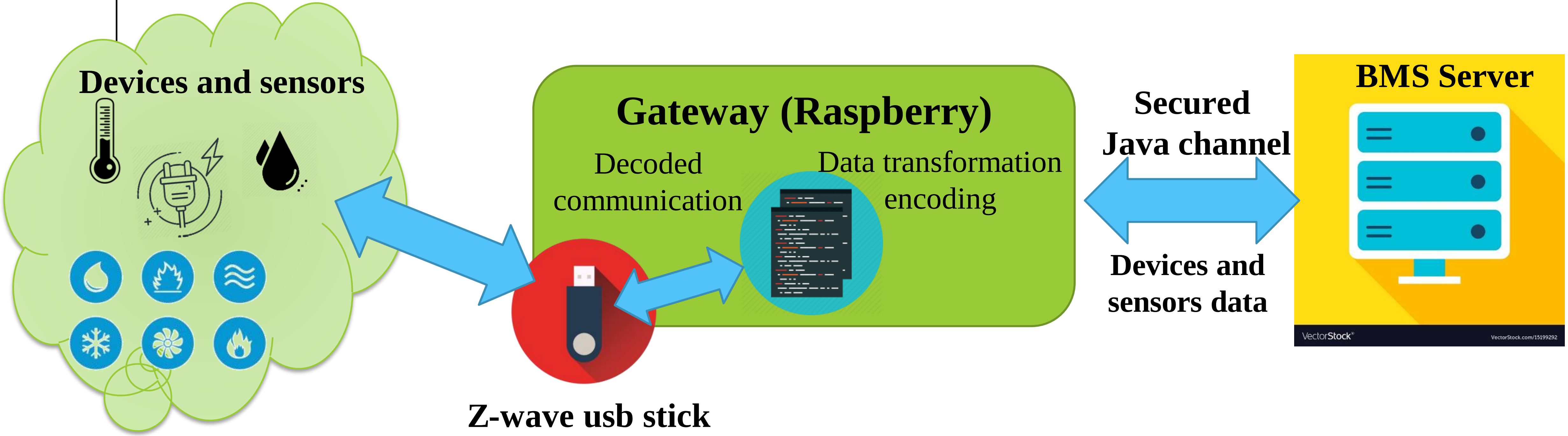
- In addition to the previously presented, we highlight
  - Secure Java Application: Java application to be used in the different Gateways to securely publish information and request actuation data into the BMS.
  - Secure Token Exchange: Python Implementation of two pieces of software:
    - Secure Token Transmitter: Entity that sends information to be ciphered with CP-ABE and distributed to the receivers.
    - Secure Token Receiver: Entity that is subscribed to the BMS, receives Secure tokens and deciphers them to perform other actions with that information.

Plug-N-Harvest: Installation of SecureJavaApp in IoT-Gateway

- ❑How to install the Secure Java Application
  - ❑1 - Flashing Raspbian Image into a microSD Card
  - ❑2 – Installing Java OpenJDK 1.8 into the Raspbian system (stable version tested)
  - ❑3 – Copy the Secure Java Application Software into the system
  - ❑4 – Install Eclipse or other Java IDE to help the process of testing and integration
  - ❑5- Create Java project
    - ❑5.1 – Import existing code < Secure Application Software
    - ❑5.2 – Add external jar libraries to the Build Path from the lib folder
  - ❑6 – Run Example IoTGateway.java to verify that it is working OK.
  - ❑7 – **Adapt Source Code of IoTGateway.java regarding CERTH gateway-software for managing sensors and actuators of pilots.**

Plug-N-Harvest: Gateways Management Framework [CERTH]

Plug-N-Harvest: Gateway



Plug-N-Harvest: Data push to BMS Server (PC & RasPi) Requirements for Secure JAVA API

❑Linux (Ubuntu)

❑Java OpenJDK-8 (Not available on Windows)

❑Python >= 3.6

❑Eclipse (latest) → only for configuration of the software (not necessarily installed on the Raspberry)

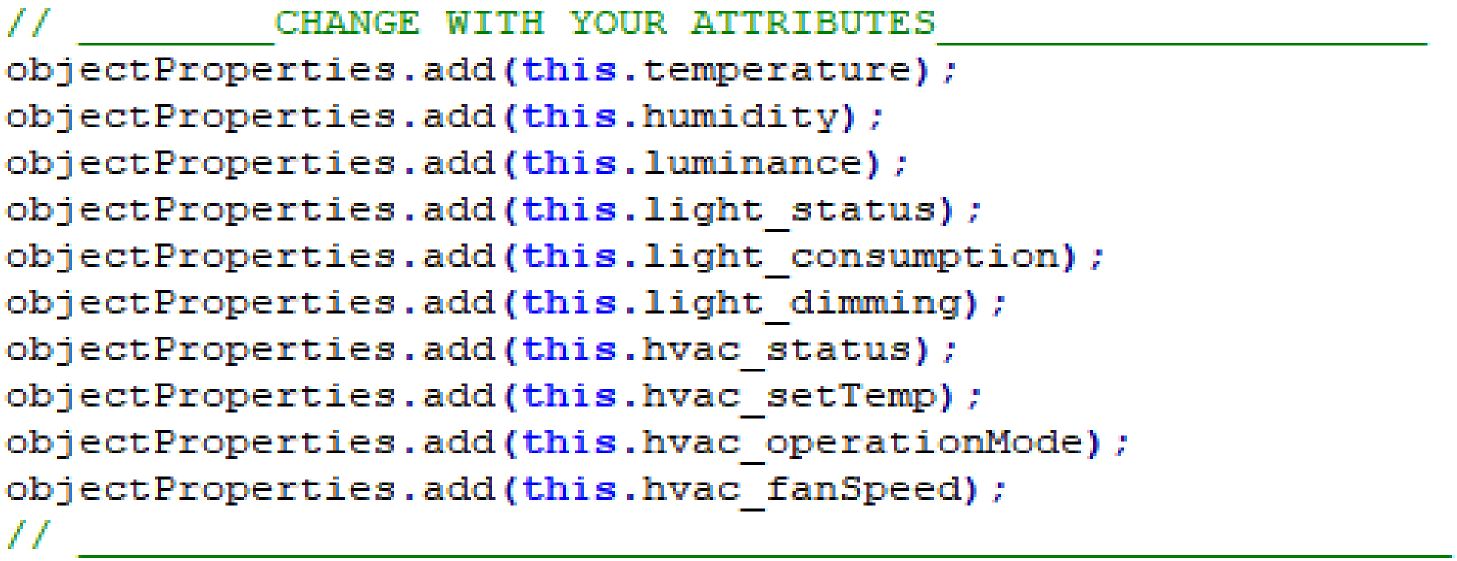
***Remark*:** The equivalent of OpenJDK-8 for Windows does not work properly with the application. Errors occur while executing IoTGateway.java. As a result, the procedure may not be completed in a Windows environment.

Plug-N-Harvest: Code configuration

- ❑Install and Configure OpenJDK-8
- ❑ import the Java app project (PnH-GatewayDemo) - OdinS
- ❑Solve library dependencies during the parameterization after the installation (see manual for more details)

Plug-N-Harvest: Set attributes to the Gateway.

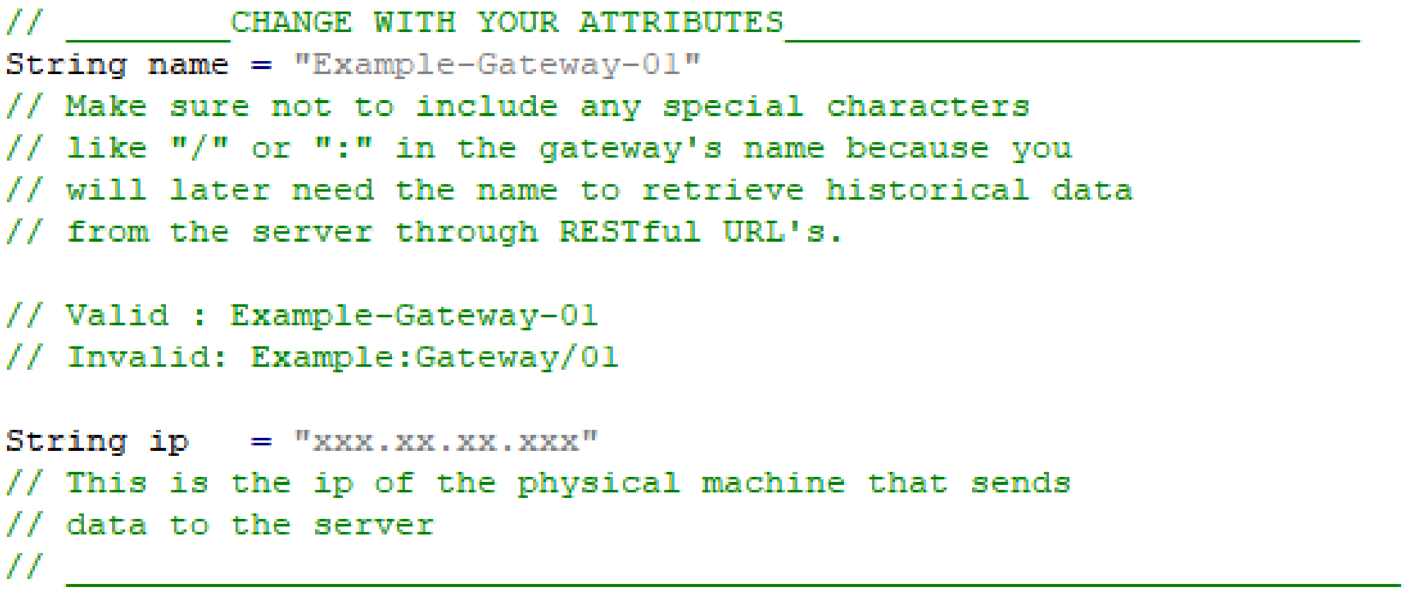
- Class IoTGateway (): Set the attributes of your gateway (e.g. humidity)



```
// ________CHANGE WITH YOUR ATTRIBUTES___________________
objectProperties.add(this.temperature);
objectProperties.add(this.humidity);
objectProperties.add(this.luminance);
objectProperties.add(this.light_status);
objectProperties.add(this.light_consumption);
objectProperties.add(this.light_dimming);
objectProperties.add(this.hvac_status);
objectProperties.add(this.hvac_setTemp);
objectProperties.add(this.hvac_operationMode);
objectProperties.add(this.hvac_fanSpeed);
// ______________________________________________________
```

Plug-N-Harvest: Gateway’s name and device IP

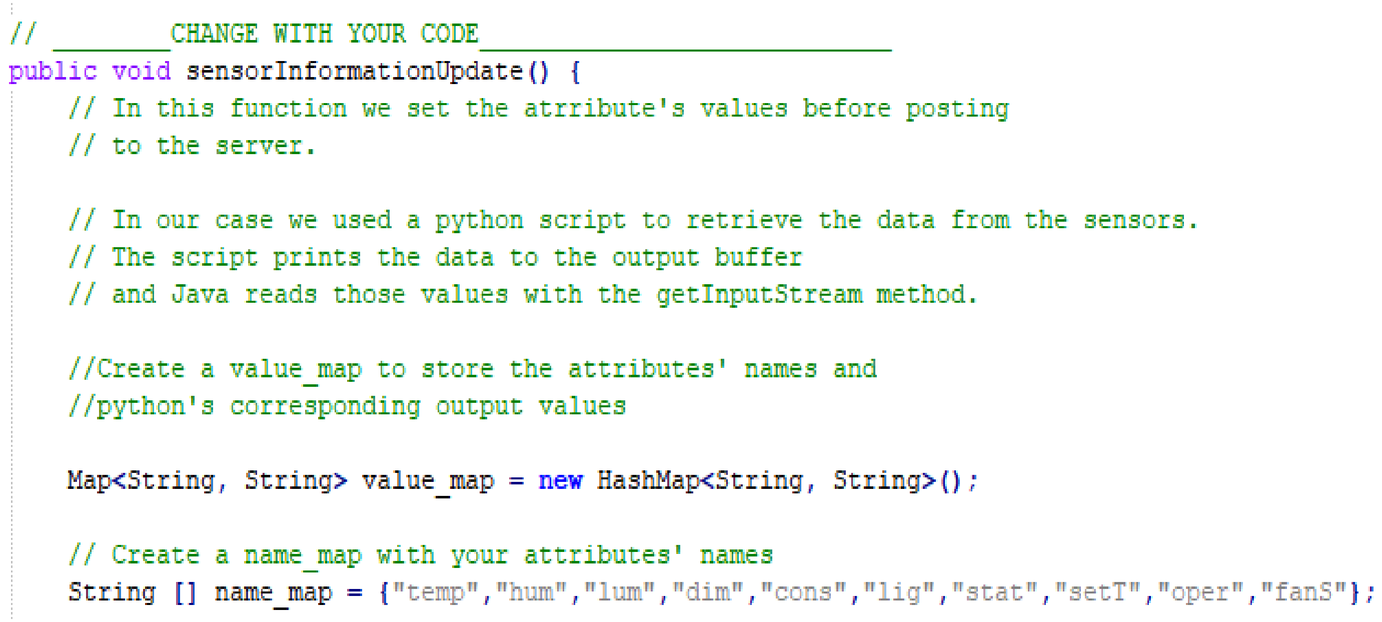
- Class IoTGateway deviceRegistration().Set the Gateway's name and the IP of the device that posts the data to the server



```
// _________CHANGE WITH YOUR ATTRIBUTES_____________________________
String name = "Example-Gateway-01"
// Make sure not to include any special characters
// like "/" or ":" in the gateway's name because you
// will later need the name to retrieve historical data
// from the server through RESTful URL's.

// Valid : Example-Gateway-01
// Invalid: Example:Gateway/01

String ip   = "xxx.xx.xx.xxx"
// This is the ip of the physical machine that sends
// data to the server
// _________________________________________________________________
```

Plug-N-Harvest: Gateway updates sensor data in BMS server

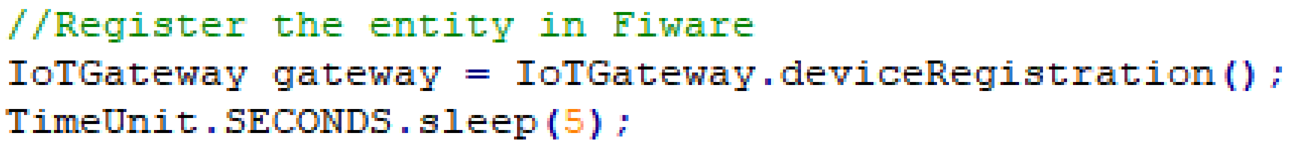
- ❑Class sensorInformationUpdate(). Update the attributes' values with the sensors data (same order with the definition)



```
//  ________CHANGE WITH YOUR CODE____________________________
public void sensorInformationUpdate() {
    // In this function we set the atrribute's values before posting
    // to the server.

    // In our case we used a python script to retrieve the data from the sensors.
    // The script prints the data to the output buffer
    // and Java reads those values with the getInputStream method.

    //Create a value_map to store the attributes' names and
    //python's corresponding output values

    Map<String, String> value_map = new HashMap<String, String>();

    // Create a name_map with your attributes' names
    String [] name_map = {"temp","hum","lum","dim","cons","lig","stat","setT","oper","fanS"};
```

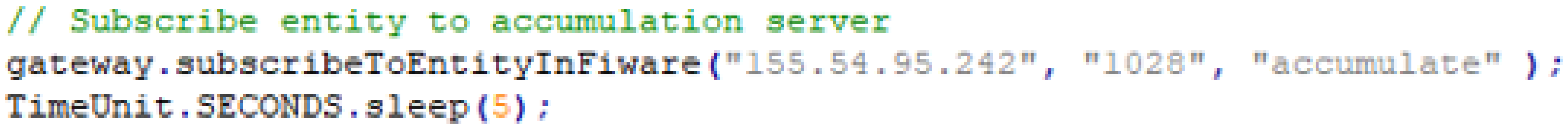
Plug-N-Harvest: Register gateway to Fiware

- Class main(). Registry of the Gateway in Fiware and Subscription Gateway to accumulation server



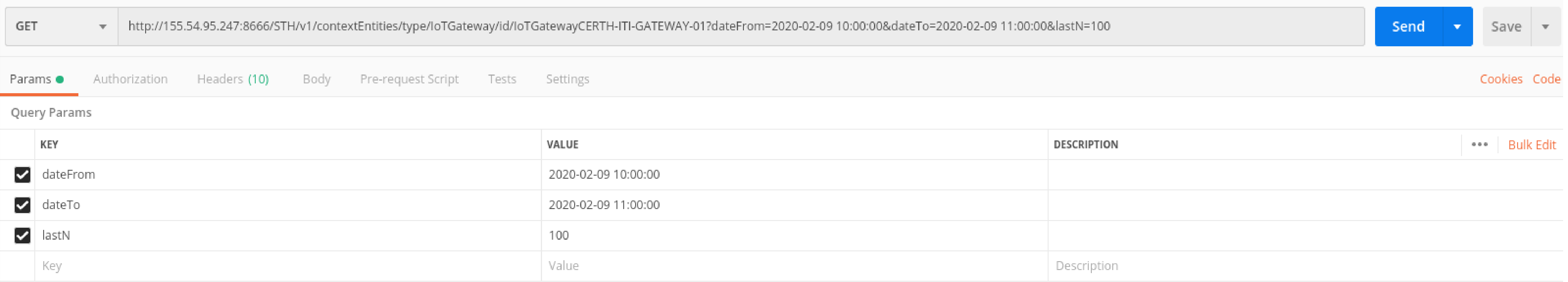
```
//Register the entity in Fiware
IoTGateway gateway = IoTGateway.deviceRegistration();
TimeUnit.SECONDS.sleep(5);
```

Subscribe Gateway to accumulation server



```
// Subscribe entity to accumulation server
gateway.subscribeToEntityInFiware("155.54.95.242", "1028", "accumulate" );
TimeUnit.SECONDS.sleep(5);
```

Plug-N-Harvest: Historical data collection module from BMS server REST API



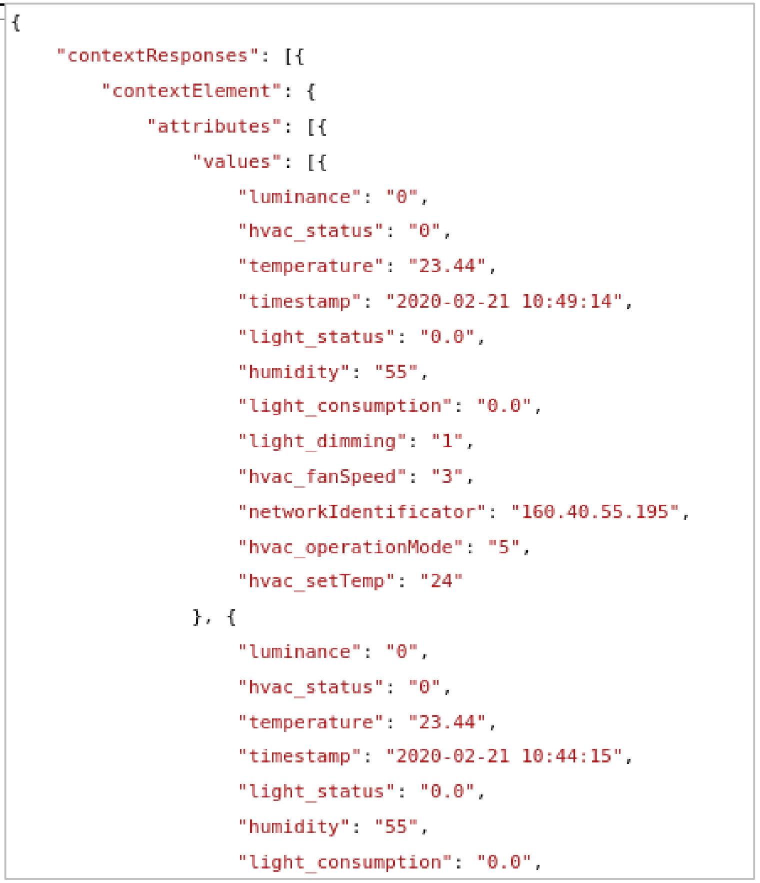
GET http://155.54.95.247:8666/STH/v1/contextEntities/type/IoTGateway/id/IoTGatewayCERTH-ITI-GATEWAY-01?dateFrom=2020-02-09 10:00:00&dateTo=2020-02-09 11:00:00&lastN=100

	KEY	VALUE	DESCRIPTION
☑	dateFrom	2020-02-09 10:00:00	
☑	dateTo	2020-02-09 11:00:00	
☑	lastN	100	
	Key	Value	Description

http://<URL>:8666/STH/v1/contextEntities/type/<Type>/id/<Id>/attributes/Conductivity?lastN=3&dateFrom=2018-09-01T16:00:00.000Z&dateTo=2020-12-31T17:00:00.000Z

- Parameters:
  - **LastN**: The requested last entries are returned. (mandatory)
  - **dateFrom**: start date and time which the raw information is returned.

Plug-N-Harvest:Historical data collection historical date example response



```
{
    "contextResponses": [{
        "contextElement": {
            "attributes": [{
                "values": [{
                    "luminance": "0",
                    "hvac_status": "0",
                    "temperature": "23.44",
                    "timestamp": "2020-02-21 10:49:14",
                    "light_status": "0.0",
                    "humidity": "55",
                    "light_consumption": "0.0",
                    "light_dimming": "1",
                    "hvac_fanSpeed": "3",
                    "networkIdentificator": "160.40.55.195",
                    "hvac_operationMode": "5",
                    "hvac_setTemp": "24"
                }, {
                    "luminance": "0",
                    "hvac_status": "0",
                    "temperature": "23.44",
                    "timestamp": "2020-02-21 10:44:15",
                    "light_status": "0.0",
                    "humidity": "55",
                    "light_consumption": "0.0",
```

Thank you!