BE in CPPS

Innovation Action Project

HORIZON 2020 - EU.2.1.5.-Ref. 680633

D3.4 – Digital World BEinCPPS Components

Lead Author: Mauro Isaja (ENG)

With contributions from: Antonio Scatoloni (ENG), Massimiliano Alberti (ENG), Nenad Stojanovic (NISSA), Antonio Jara (HOPU)

Deliverable characteristics

Deliverable nature	O
Dissemination level	PU
Contractual delivery date	31 October 2017
Actual delivery date	30 November 2017
Version	1.0
Keywords	

Version history

Nr.	Date	Notes and comments
0.1	12/09/2017	ToC
0.2	20/10/2017	Main body of the document
0.3	24/11/2017	HOPU contributions.
0.4	29/11/2017	NISSA contribution
0.9	29/11/2017	First complete version shared internally
1.0	30/11/2017	Final version ready for release

Executive summary

This D3.4 deliverable is the second and final deployment of the BEinCPPS software components belonging to the *Digital World* (DW) – i.e., the IT infrastructure supporting the execution of "Cyber" side of Cyber-Physical Production Systems. The present document reports about updates to the Common Cloud Environment (CCE) and about the online availability of Value Added Services (VAS). Only relevant changes with respect to the first release have been reported here: for information of DW assets not reported here and on the technical characteristics of the CCE infrastructure itself, please refer to the already released documentation (D3.3).

The CCE is a *sandbox* execution environment that was set up by the BEinCPPS consortium for internal testing purposes and for providing a fully-functional instance of the BEinCPPS Platforms to external developers. It hosts the full range of DW components. With respect to the first release, the following updates have been applied:

- Upgrade of the FIWARE Orion Context Broker GE;
- Deployment of the FIWARE Cygnus Connector GE;
- Update of the 3D Visualization Services and Asset Registry for CPPS;
- Deployment and integration of the FIWARE security layer.

On the other hand, VAS are commercial software delivered in SaaS mode, as opposed to the open source nature of the building blocks of the BEinCPPS platforms. There are two VAS assets that, being in the scope of DW and a novelty with respect to the previous version, are reported here:

- D2Lab CPPS Analysis
- Homard IoT Management for CPPS

Each of these bullet points is discussed in some detail in a dedicated section of the document.

Table of contents

1 Introduction 6
1.1 Scope and purpose of this deliverable 6
1.2 Organization of this document 7
2 Updates to the Common Cloud Environment 8
2.1 FIWARE Orion Context Broker 11
2.2 FIWARE Cygnus Connector 11
2.3 FIWARE KeyRock Identity Management 12
2.4 FIWARE Wilma Policy Enforcement Proxy 13
2.5 Asset Registry for CPPS 14
2.6 3D Visualization Services 14
3 Value Added Services and Applications 15
3.1 D2Lab CPPS Analysis 15
3.2 Data Upload and Detection API 15
3.3 Training and detection data format 20
3.4 Homard IoT Management for CPPS 22

List of Figures

Figure 1 - DW components of the BEinCPPS Platform 6
Figure 2 - Value Added Services in the BEinCPPS context 7
Figure 3 - Deployment diagram of the CCE 10
Figure 5 - Integration of FIWARE Cygnus 12
Figure 6 - Integration of FIWARE KeyRock 13
Figure 7 - Integration of FIWARE Wilma 13
Figure 8 - Integration of AR4CPPS 14
Figure 9 - Deployment of 3DVS 14
Figure 10: Homard Architecture 22
Figure 11: Task Orchestration for CPPS enviroment 23
Figure 12: Homard Login View 24
Figure 13: Homard Dashboard 24
Figure 14: Homard Device Management 25
Figure 15: Network Health Monitor 26

1 Introduction

1.1 Scope and purpose of this deliverable

Deliverable D3.4 is the second and final deployment of the BEinCPPS software components belonging to the *Digital World* (DW) – i.e., the IT infrastructure supporting the execution of "Cyber" side of Cyber-Physical Production Systems. The DW is where Real World entities are digitally represented and shopfloor data is stored and processed. As explained in the documentation of the first release of this deliverable (D3.3, M6), DW components are those belonging to the Cloud and (partially) Factory levels of the BEinCPPS architecture, as shown in Figure 1 below.

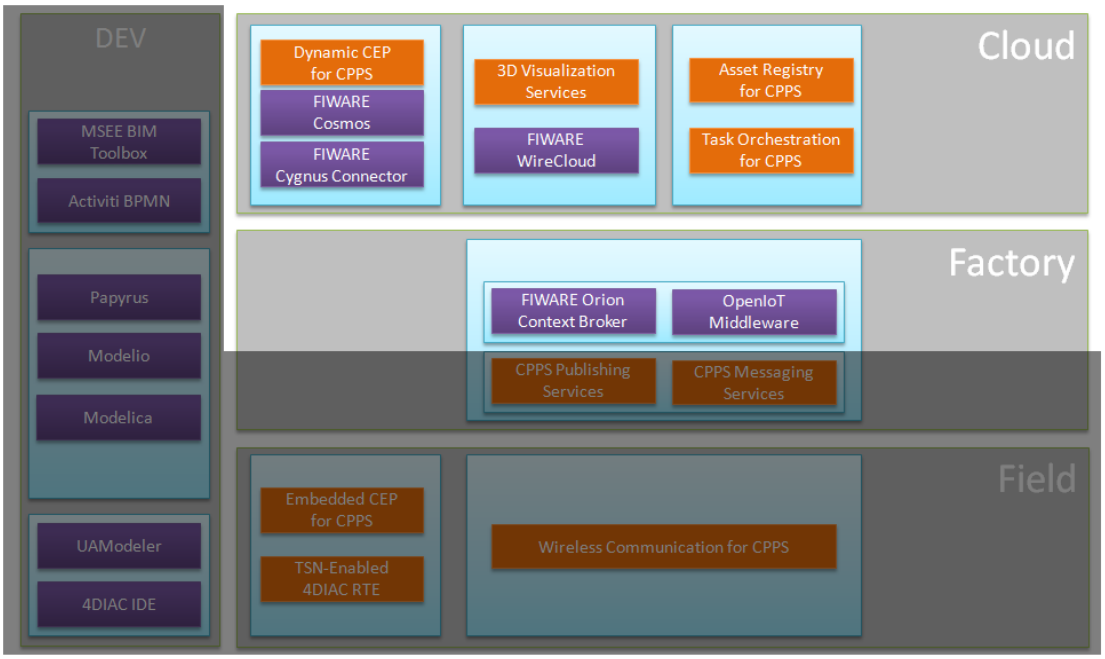

Figure 1 - DW components of the BEinCPPS Platform

The picture above represents a full view on the three BEinCPPS Platforms[1], which support *runtime* operations, and on the *modelling, design, engineering and development* tools (the DEV box) supported in the BEinCPPS ecosystem. In particular, all software components belonging to any of the Platforms are open source; as such they are available online for free use and their code can also be modified and redistributed in accordance to the licensing terms[2].

On the other hand, Value Added Services (VAS), while being foreground BEinCPPS assets, are considered as external components with respect to the

[1] Generally speaking, FI Platform = Cloud Level, IoT Platform = Factory Level and SS Platform = Field Level; however, some minor exceptions to this rule exist (e.g., Task Orchestration for CPPS being positioned on the Cloud level but still belonging to the IoT Platform), and for a more clear definition of the three scopes please refer to WP2 deliverables.

[2] Link for downloading the source distribution of every component are provided in the documentation of WP2 deliverables.

BEinCPPS Platforms: they are commercial software that is delivered in SaaS mode, as opposed to the open source nature of the building blocks of the BEinCPPS Platforms. The VAS items that are in DW scope are those highlighted in Figure 2 below, adapted from deliverable D2.2. For two of them, namely *D2Lab CPPS Analytics* and *Homard IoT Management for CPPS*, a dedicated subsection describes the technical details of their deployment. The other two have been omitted for different reasons: the deployment of *i-LiKe Machines CPPS Monitor* is actually unchanged with respect to the first release, while *Time-Sensitive Network for CPPS* is a hardware-based asset that is in RW scope, and as such its deployment is reported in deliverable *D3.2 Real World BEinCPPS Components*.

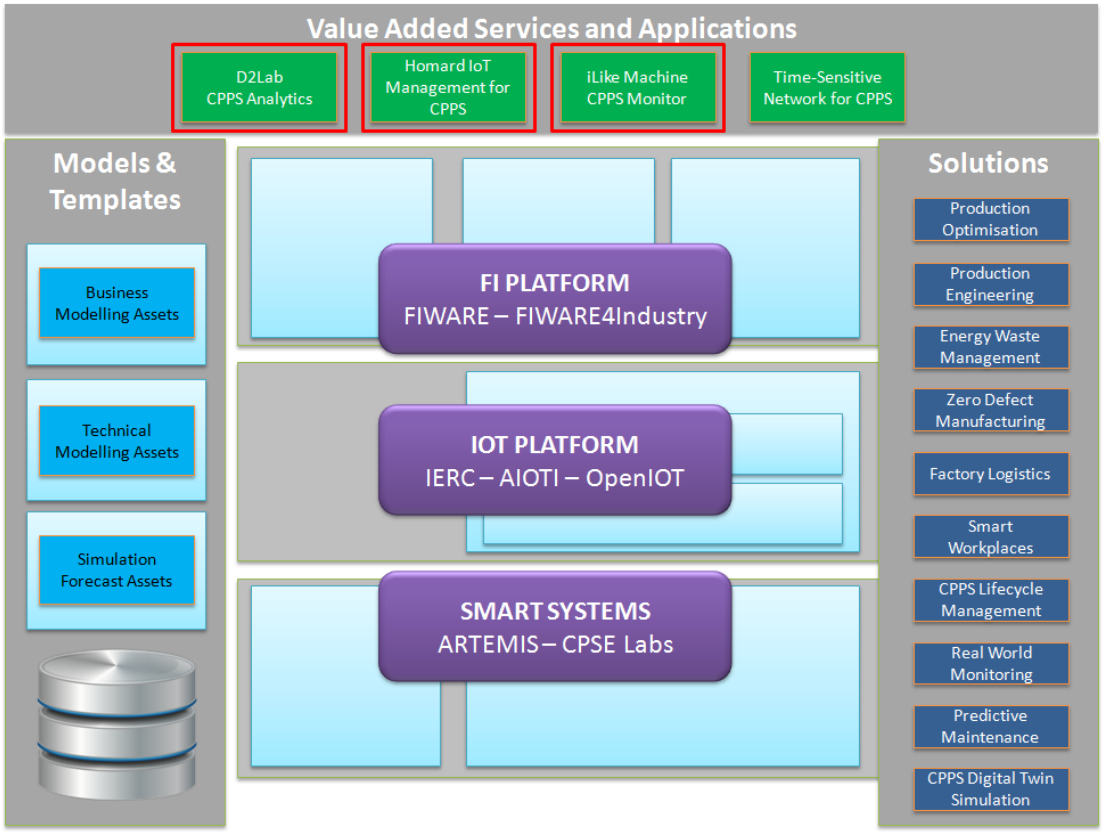

Figure 2 - Value Added Services in the BEinCPPS context

1.2 Organization of this document

This document reports about the deployment of DW components during the second iteration of the WP3 technical tasks – i.e. M7-M24. It is composed of two parts: §2 describes the updates to the Common Cloud Environment (CCE), while §3 is about the online availability of Value Added Services (VAS). **Only changes with respect to the first release have been reported in this document; for information on DW assets not described here and on the technical characteristics of the CCE infrastructure itself, please refer to deliverable D3.3.**

2 Updates to the Common Cloud Environment

As already explained in the first release of this deliverable, the Common Cloud Environment (CCE) is a *sandbox* execution environment that was set up by the BEinCPPS consortium for internal testing purposes and for providing a fully-functional instance of the BEinCPPS Platforms to external developers – e.g., winners of the two BEinCPPS Open Calls. It hosts the full range of BEinCPPS Digital World components, which have been virtualized by leveraging the Docker[3] platform: this way, it is very easy to replicate the CCE or a selection of individual components in different environments.

We refer to the CCE as a single logical entity, although it is composed by some independent computing infrastructures under the umbrella of the <REMOVED FROM PUBLIC VERSION> domain. However, all the updates of the second release impacted on one specific computing infrastructure: the commercial service provided by OVH[4], which is physically located in the Strasbourg area and is reachable through the <REMOVED FROM PUBLIC VERSION> network address.

Figure 3 shows the complete picture of the CCE as a deployment diagram. All components are boxed within their own **Docker container instance**, sometimes shared with some auxiliary software module that is not considered as a DW component by itself (e.g., the NGSI Proxy and the FiVES Synchronization Server).

Public endpoints exposed by components are listed vertically on the left hand side, with their respective TCP/IP port number[5]. These endpoints expose either a web-based user interface (UI) or a web-based service interface (API); when a standard protocol is used for communication, this is also mentioned (e.g., NGSI, HTML, AMQP).

On the other hand, endpoints used for **internal integration** are easily identified in the diagram as they do not declare any port number[6] and are (most of the times) linked by other components that depend on them[7]. More details on integration points are given in the sections dedicated to individual components. However, it is worth noting that there are basically three independent subsystem without integration points bridging them: one FIWARE-based, one OpenIoT-based plus the standalone 3D Visualization Services (3DVS) component

In the FIWARE-based subsystem, three components provide back-end services to the others: FIWARE Cygnus Connector, which saves live data to a MySQL relational database, FIWARE KeyRock providing Identity Management (IDM) functionality and FIWARE Orion Context Broker (OCB) that implements a publish/subscribe

[3] https://www.docker.com/
[4] https://www.ovh.co.uk/
[5] E.g., the AR4CPPS component can be reached at the address <REMOVED FROM PUBLIC VERSION>
[6] Actually, the CCE firewall is barring by policy any external incoming connection to internal interfaces.
[7] E.g., the FIWARE OCB component is integrated with four other components.

databus based on the NGSI standard. OCB, in turn, is used by Asset Registry for CPPS (AR4CPPS) and by FIWARE Wirecloud, but its services are also exposed on the public Internet through the FIWARE Wilma Policy Enforcement Proxy (PEP), which restricts the use of OCB's NGSI API to authorized users.

The OpenIoT-based subsystem is much simpler from an integration perspective, with the Virtuoso Server providing *internal* data persistence services to the RabbitMQ messaging middleware. RabbitMQ, besides exposing AMQP and MQTT-over-WebSockets functionality on the public Internet, also has internal integration points for peer discovery and command-line tools (shown on its right hand side) that are currently not used.

Finally, the 3DVS component stand by itself in that it only provides internal integration with the FiVES Synchronization Server.

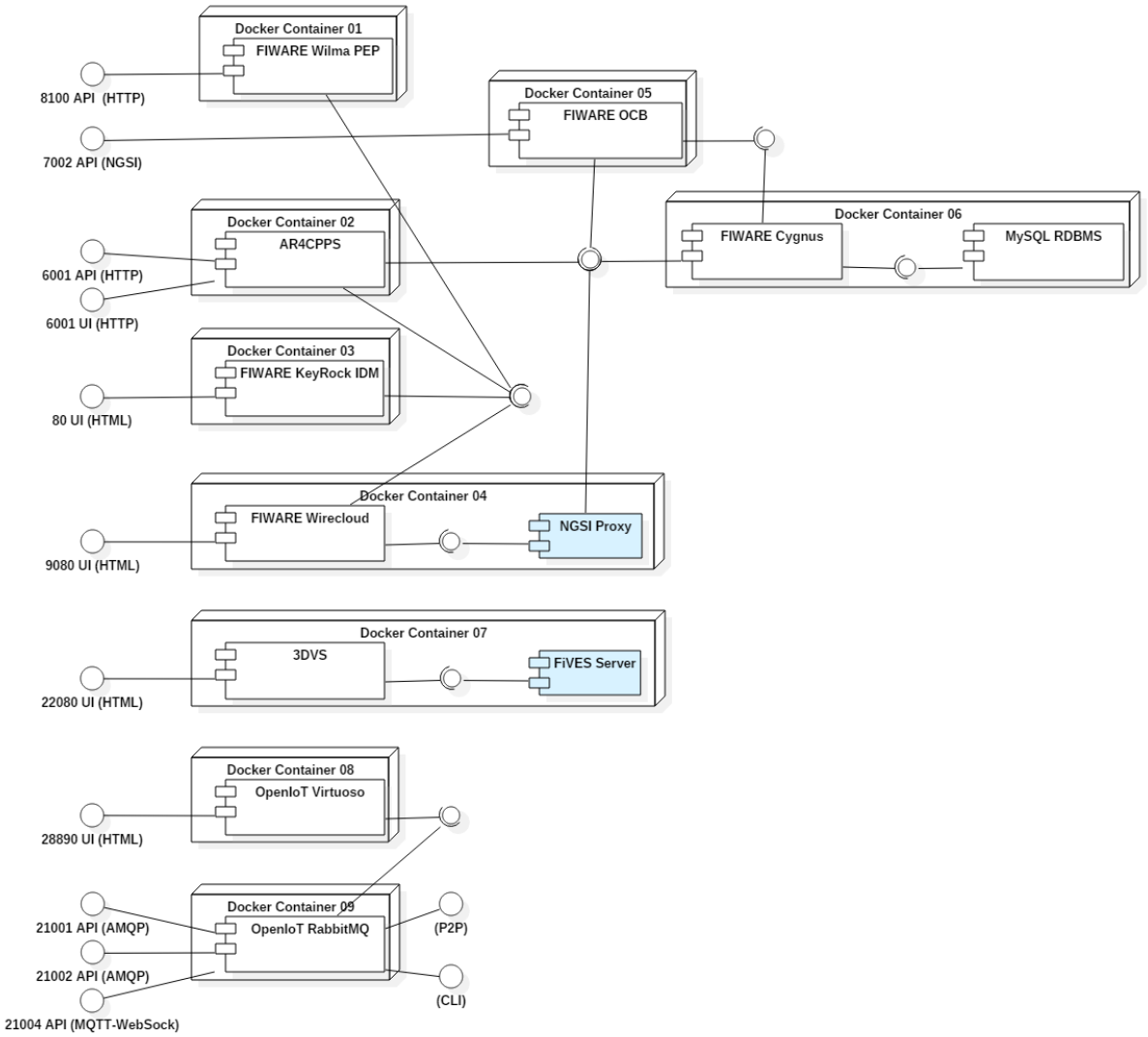

Figure 3 - Deployment diagram of the CCE

With respect to the first release, the following updates have been applied to the CCE:

- Update of the FIWARE Orion Context Broker GE (OCB) instance to the latest version available from the community site;
- Deployment of a data persistence service for OCB, by means of the FIWARE Cygnus GE;
- Update of the 3D Visualization Services (3DVS) and Asset Registry for CPPS (AR4CPPS) installations to the latest versions released by the BEinCPPS project (see deliverable D2.6);
- Deployment and integration of the FIWARE security layer, based on the KeyRock IDM (identity management) and Wilma PEP (policy enforcement proxy) GEs.

The last point in particular represents the most important added value of this release, as it closes the previously existing gap between the CCE and a fully functional cloud infrastructure as can be used for the delivery of commercial-grade solution. In the following sub-sections, each of these bullet points is described in detail.

Overall, the CCE is an infrastructure ready for third-party use: external access to the public endpoints and to individual CCE modules packaged as Docker images (see Figure 3) can be requested by contacting the CCE admin team at the following email address:

- rdlabeng@gmail.com

2.1 FIWARE Orion Context Broker

The new version of the OCB installed on the CCE is 1.9.0, released on 19 October 2017. It introduces several improvements and a huge number of fixes over the previously installed 1.0.0 version. The most notable improvements are:

- Mature implementation of the NGSIv2 API, including support for attribute blacklist in notifications and for georeferenced context data
- Improved query language for context metadata
- Support for "permanent" subscriptions
- Support for notifications over HTPPS secure channels provided by the subscribers, with the additional option of accepting self-signed certificates (a common setup for internal integration)
- Introduction of a REST API for metrics

Note that v1.9.0 supports both NGSIv1 and NGSIv2, and the latter is recommended version as it is actively maintained. That said, NGSIv2 still lacks functionality for context discovery (also known as NGSI9, as opposed to NGSI10 which defines the publish/subscribe interface for context data), so that clients will still have to use NGSIv1 if context discovery is required. Note also that XML data is now deprecated and support for it has been removed.

Both NGSIv1 and NGSIv2 are online at the address `<REMOVED FROM PUBLIC VERSION>`. To use this public endpoint, however, clients should call the FIWARE Wilma PEP passing an OAuth2.0 token obtained from the FIWARE KeyRock IDM (see the relevant section below): it the token is valid, the call will be redirected to the OCB endpoint.

2.2 FIWARE Cygnus Connector

The FIWARE Cygnus Connector is a new entry in CCE, introduced as the result of lessons learned from the first BEinCPPS development cycle. Cygnus meets the requirement, which arose from some of the BEinCPPS pilots, of a persistence

mechanism that may turn a live data stream into an historical (and searchable) database, without any additional programming. On the one side, Cygnus subscribes to a live context using the NGSI endpoint provided by OCB (subscription) and its own internal endpoint (incoming notifications); on the other, it forwards all received data to a MySQL database instance.

As no public endpoint is exposed, the Cygnus component is actually not part of the CCE external interface. Internally, the two endpoints exposed by Cygnus are `<REMOVED FROM PUBLIC VERSION>` and `<REMOVED FROM PUBLIC VERSION>`.

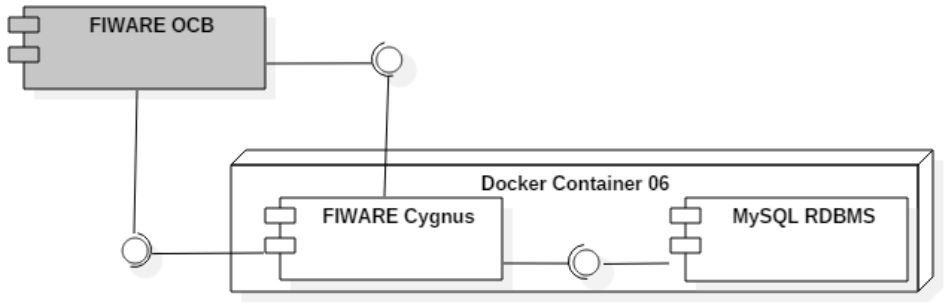

Figure 4 - Integration of FIWARE Cygnus

2.3 FIWARE KeyRock Identity Management

The FIWARE KeyRock Identity Management (IDM)[8] is also a new installation for the CCE. It is derived from OpenStack KeyStone[9]. It implements a Single-Sign-On (SSO) protocol based on the OAuth2.0 standard. In the context of the CCE, IDM is the component in charge of authentication, authorization and user profile management for the entire FIWARE-based subsystem. In particular, access to APIs that are protected by the Wilma PEP (see section below) requires the callers to first authenticate themselves with the IDM and obtain a OAuth2.0 token. Moreover, both AR4CPPS' and Wirecloud's web UIs use the SSO service provided by IDM to authenticate their users.

On the public Internet, the web-based admin interface of IDM can be reached at the address `<REMOVED FROM PUBLIC VERSION>`. Internally, components integrate with the IDM through its *private* endpoint[10] at the address `<REMOVED FROM PUBLIC VERSION>`.

[8] https://catalogue.fiware.org/enablers/identity-management-keyrock
[9] https://docs.openstack.org/keystone/pike/
[10] I.e., not reachablefrom the public Internet

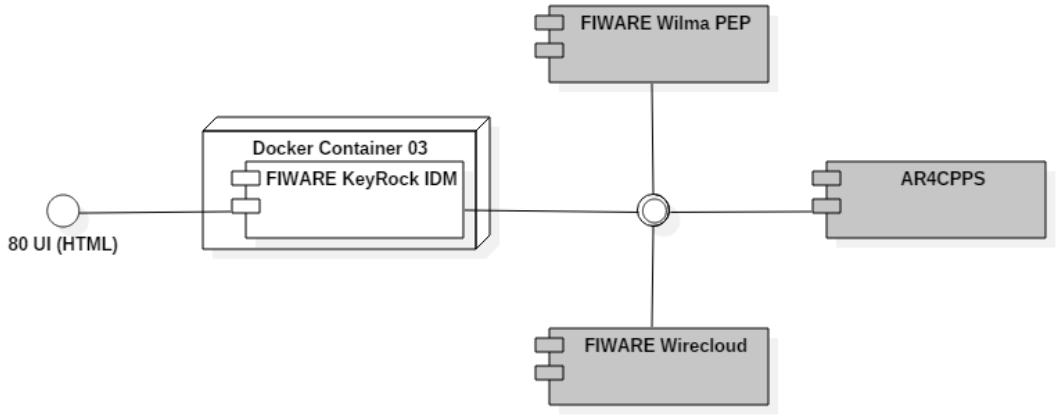

Figure 5 - Integration of FIWARE KeyRock

2.4 FIWARE Wilma Policy Enforcement Proxy

The FIWARE Wilma Policy Enforcement Proxy (PEP)[11] complements the IDM and provides OAuth2.0 token validation services for enforcing access restriction on third-party APIs. In the CCE context, PEP functionality is used to protect the public OCB endpoint from unauthorized access. OCB clients will first obtain a valid OAuth2.0 token from IDM, make their call to the public PEP endpoint at the address `<REMOVED FROM PUBLIC VERSION>`, and get finally redirected to the actual OCB endpoint.

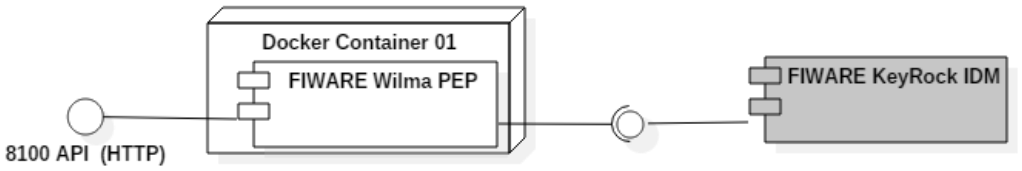

Figure 6 - Integration of FIWARE Wilma

[11] https://catalogue.fiware.org/enablers/pep-proxy-wilma

2.5 Asset Registry for CPPS

The Asset Registry for CPPS (AR4CPPS) is a foreground component of the BEinCPPS project. As its name suggests, it implements a registry of digital records that represent physical devices on the shopfloor – for more details, please refer to the D2.6 deliverable. AR4CPPS was already present in the first installation of the CCE, but the new version has introduced the integration with OCB and IDM.

In the CCE, both the UI and the API endpoints are exposed on the public Internet at the addresses `<REMOVED FROM PUBLIC VERSION>` and `<REMOVED FROM PUBLIC VERSION>`, respectively. On the back-end side, AR4CPPS integrates with the public NGSI endpoint of OCB and with the private OAuth2.0 endpoint of IDM. It is worth noting that access control policies, based on IDM identities and roles, are directly enforced by the AR4CPPS' API implementation, so that no PEP is required.

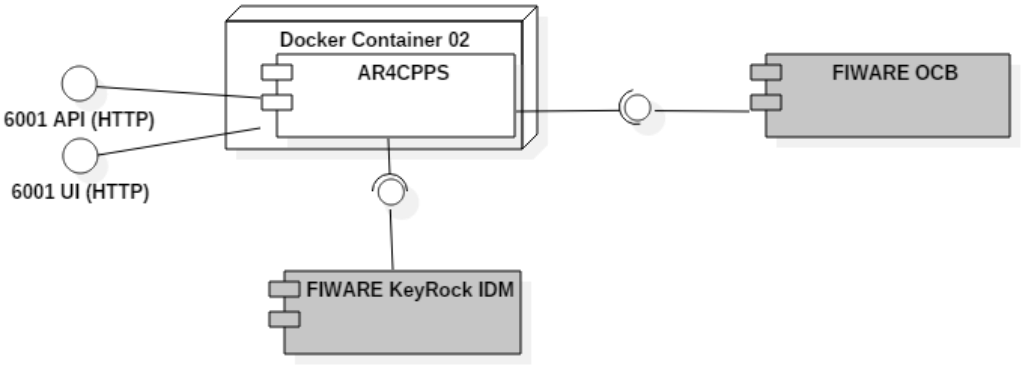

Figure 7 - Integration of AR4CPPS

2.6 3D Visualization Services

The 3D Visualization Services (3DVS) is also reported in this new release as the software went through a number of improvements, which are documented in the D2.6 deliverable. From the integration point of view, no changes have been made.

In the CCE, the UI of 3DVS is exposed on the public Internet at the address `<REMOVED FROM PUBLIC VERSION>`.

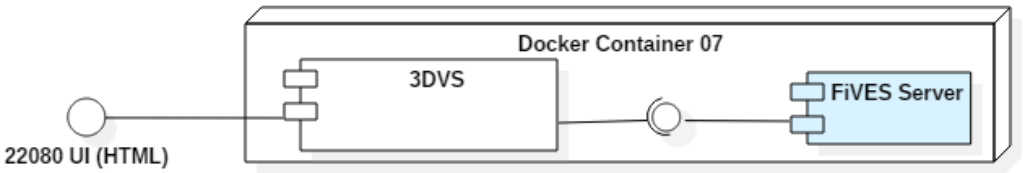

Figure 8 - Deployment of 3DVS

3 Value Added Services and Applications

3.1 D2Lab CPPS Analysis

In this section we provide details related to the usage of D2Lab. It is based on the Dynamic CEP for CPPS component described in deliverable D2.6[12].

D2Lab provides machine learning/anomaly detection as a service. More precisely, it provides real-time anomaly detection. Overall, there are seven necessary phases for successful service usage and connection with industry systems:

1. Creating project (one project is created for each product model)
2. Configuring project and training (setting up various parameters for data preprocessing and training process)
3. **Uploading training files**
4. Periodically creating models based on training data
5. **Uploading files for detection**
6. Configuring detection (setting up parameters for detection process e.g. algorithm and tolerance)
7. **Real-time detection**

This document will cover steps 3, 5 and 7. Other steps will be handled by the service providers. These phases are depending on output parameters from potentially multiple previous phases. All phases except phase 1 are recurring during project lifecycle. However, parameters set during project creation can be altered if needed.

3.2 Data Upload and Detection API

Prerequisites

Prerequisites are necessary actions done by service provider to get system ready for using. These actions are contained in phases 1, 2 and 4 and 6. This section shortly explains these actions for better service functionalities understanding and simpler error reporting. In order to start phase 3, 5 and 7, project needs to be created (phase 1), configured (phase 2) and at least one training must be done (phase 4).

Project creation includes specifying details for each parameter e.g. their threshold values (minimum, maximum), which are later used for data filtering. Details for all parameters must be specified.

Configuring project through phases 2 and 6 represents specifying training and detection algorithms and fine tuning them using various parameters. Multiple training and detection configurations may be present but only two are active at the same time – one for training and the other for detection.

[12] This is also the reason why no instance of the CEP component was installed in the CCE.

Training is complex, iterative operation and it's execution time can be in some cases long, but on the other hand it's not needed to be done frequently because model changes are minor when most of the data is same. Training is scheduled and done periodically.

Once again, please note that before real-time detection starts, there must be at least one training done.

Uploading files for training (Phase 3)

Training data represents a set of files that is uploaded in JSON format using HTTP POST request described below. One file is uploaded per request. A subset of training data is used for training based on time window specified in active training configuration. Only the data that belongs to that time window is used. Format of the file will be presented in the last section.

Request description for training file upload:

POST /projects/{projectID}/files/training

Parameters:

Name	Located in	Description	Required
projectID	path	It is provided after project is created	YES
file	formData	JSON formatted file	YES

Responses

Code	Description	Schema

202	File accepted for processing	**trainingFileCallback**{
		id: long
		Callback URI to the newly uploaded training file
		uri: string
		}

404	Project with given ID not found	

Response contains URI for checking file status and file ID number. Status can be: *QUEUED, STORED, PROCESSING* and *ERROR*.

QUEUED *– file is enqueued and waiting to be processed*

PROCESSING *– file is being processed according to project setup*

STORED *– file is stored and ready to be used*

ERROR *– file is rejected due to corrupted parameter values*

Uploading files for detection (Phase 5)

Detection data represents the data that is uploaded with the purpose of detecting abnormal behaviour. Detection is performed based on the model that was generated during training phase. Detection data is uploaded in a similar way as training data and has the same format.

Request description for detection file upload:

POST /projects/{projectID}/files/detection

Parameters:

Name	Located in	Description	Required
projectID	path	It is provided after project is created	YES
file	formData	JSON formatted file	YES

Responses

Code	Description	Schema

202	File accepted for processing	**trainingFileCallback**{
		id: long
		Callback URI to the newly uploaded training file
		uri: string
		}

404	Project with given ID not found	

Response contains URI for checking file status. Refer to previous section for possible file statuses explanations.

Real-time anomaly detection (phase 7)

After training has been done, detection configured and detection files uploaded - have status STORED, these files can be used for real time detection. Detection is done by calling HTTP POST request described below:

POST /projects/{projectID}/files/detection/{fileID}/start

Parameters

Name	Located in	Description	Required
projectID	path	It is provided after project is created	YES
fileID	path	ID of file for detection, provided as upload response	YES

Responses

Code	Description	Schema

200	Detection successful	**detectionResult**{
		id : long
		isAnomaly : boolean
		closestMedoidId : string
		distanceToClosestMedoid : double
		reason : string
		}

400	Active detection configuration missing	string

404	File does not exist	**errorResponse**{
		errors: [string]
		}

404	File not stored yet	**errorResponse**{
		errors: [string]
		}

404	Project not found	**errorResponse**{
		errors: [string]
		}

In case of successful detection, indication whether this file is anomaly can be found in boolean **isAnomaly** parameter of the JSON response. Other detection result details are also part of response message.

Detection response contains only system generated file ID. In order to see the file metadata, specific HTTP GET request for file information retrieval must be issued:

GET /projects/{projectID}/files/detection/{fileID}

Parameters

Name	Located in	Description	Required
projectID	path	It is provided after project is created	YES
fileID	path	ID of file for detection, provided in detection response	YES

Responses

Code	Description	Schema
200	Detection successful	**MadFile**{ id: long location: string who: string description: string status: string madFileType: string

		statusAttribute: string timestamp: long hBaseKey: string }
404	File not found	

3.3 Training and detection data format

Our system supports two types of data records. One is **time series** and the other is **contextual-behavioural.** This document covers time series data format.

Time series data format

As we mentioned earlier, training and detection data files have the same JSON format. This file is separated in two parts: **header** and **parameters**.

Header is JSON object which contains following key/value fields which are file metadata:

- ***timestamp***
- ***location***
- ***who***
- ***description*** (This parameter can be used arbitrarily. For example, you can set it as your internal test file ID (note that we generate new file ID for our system) to be able to identify test file after detection results are ready)

Parameters is array of objects. These objects contain parameter name and array of values. Time series file example can be found on the next page.

File example:


```
{
     "header":{
            "timestamp": "1490565600000",
            "location": "Italy",
            "who": "Federico B.",
            "description": "105-6-367666"
        }
     "parameters": [{
            "name": "Cos Fi",
            "values": [638.0, 630.0, 627.0, 635.0, 635.0, 631.0, 631.0, 632.0,
     631.0, 632.0]
            },
            {
            "name": "Temp. Sinistra (Nera)",
            "values": [631.0, 631.0, 629.0, 627.0, 627.0, 628.0, 631.0, 628.0,
     630.0, 632.0]
            },
            {
            "name": "Temp. Destra (Rossa)"
            "values": [485.0, 485.0, 486.0, 486.0, 487.0, 487.0, 487.0, 488.0,
     488.0, 489.0]
            },
            {
            "name": "T EV Cap. Fr."
            "values": [271.0, 272.0, 271.0, 272.0, 271.0, 272.0, 271.0, 272.0,
     271.0, 272.0]
            },
            {
            "name": "T EV Cap. Fz."
            "values": [270.0, 270.0, 270.0, 270.0, 270.0, 270.0, 270.0, 270.0,
     270.0, 270.0]
            },
            {
            "name": "Potenza"
            "values": [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 3.0]
            }
     }]
}
```

3.4 Homard IoT Management for CPPS

Description and Functionality

Homard serves as a management platform for OMA LwM2M devices, offering functionalities such as firmware upgrades, remote maintenance, device diagnostics, etc. OMA LwM2M is an evolution of the former OMA Device Management, which is one of the most used protocols for device management. That makes OMA LwM2M an outstanding standard with a lot of experience behind its back.

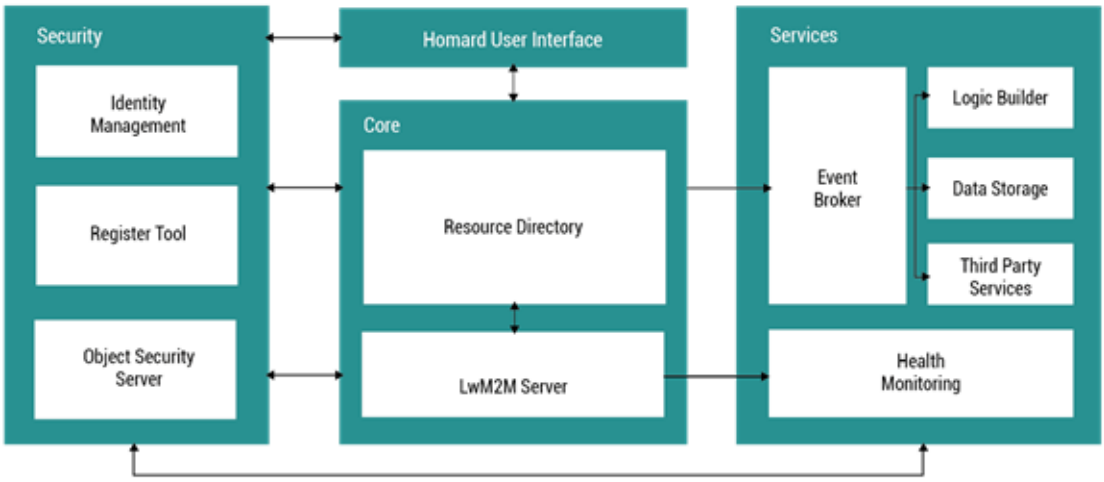

Figure 9: Homard Architecture

Homard can be divided into 4 different modules: the Core, acting as a OMA LwM2M device manager; a Security module where registration are made; a User Interface for users to interact with the platform; a Services module that serves the system of functionalities.

These are its main functionalities:

- Software Management: enabling the installation, removal of applications, and retrieval of the inventory of software components already installed on the device and the most relevant firmware upgrade over the air.
- Diagnostics and Monitoring: enabling remote diagnostic and standardized object for the collection of the memory status, battery status, radio measures, QoS parameters, peripheral status and other relevant parameters for remote monitoring.
- Connectivity and Security: allowing the configuration of bearers (WiFi, Bluetooth, cellular connectivity), proxies, list of authorized servers for remote firmware upgrade and also all the relevant parameters for enabling secure communication.
- Device Capabilities: allowing the Management Authority to remotely enable and disable device peripherals like cameras, Bluetooth, USB,

sensors (ultrasound, temperature, humidity, etc.) and other relevant peripherals from the nodes.

- Lock and Wipe: allowing to remotely lock and/or wipe the device, for instance when the device is lost (relevant for devices in open ocean, air etc.), or when the devices are stolen or sold. It enables the remote erase of personal / enterprise data when they are compromised.
- Management Policy: allowing the deployment on the device of policies which the client (node, device, sensor) can execute and enforce independently under some specific conditions, i.e., if some events happen, then perform some operations.

In order to support the Field Level devices managed by Homard, Task Orchestration for CPPS serves as a complementary tool for subscribing to events, linked via BPMs or via other REST-based API services.

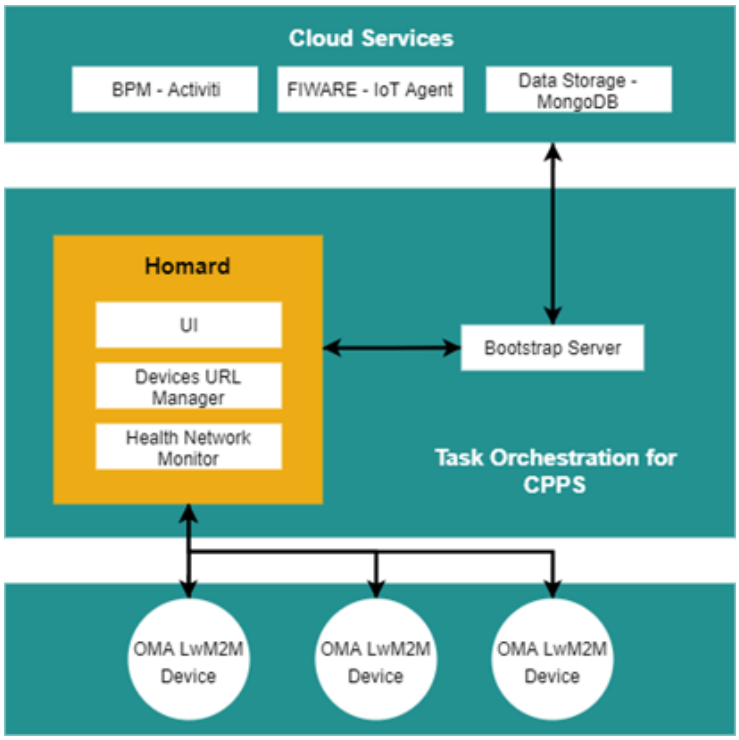

Figure 10: Task Orchestration for CPPS enviroment

The following figure shows the user interface that gives access to the Homard platform. The platform uses an authentication system based on OAuth2 that allows the registration of users in a secure way, being able to ensure that their data and permissions remain inaccessible to any attacker and even to ourselves.

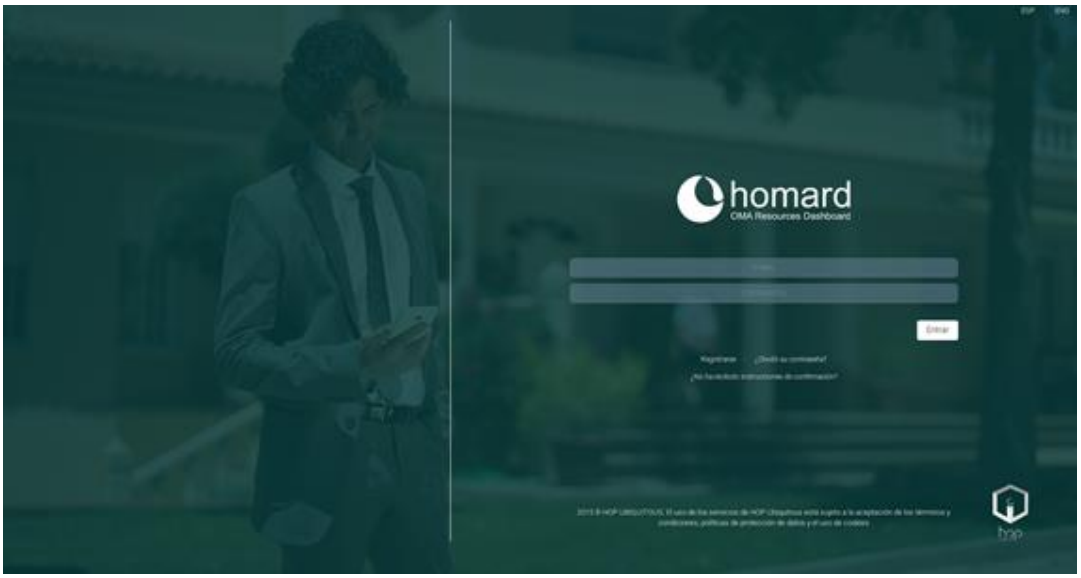

Figure 11: Homard Login View

The Homard dashboard is shown in the figure below. It allows the user to get used to the interface and available tools with a quick look. The right column displays the running devices taks and recent events, while the left one contains the device management tool, logic builder, devices online, tutorial, etc.

Figure 12: Homard Dashboard

The device management works accordingly the OMA LwM2M data modeling. That is, you can access resources of the objects available in the device registered. By that you can read, write or execute them depending on their nature. There is a first level represented by objects and within each object the set of resources that compose

it. In addition an intermediate layer is added, the interfaces, which allow multiple representations of the same object in a single device. The existence of the instances may respond, for example, to the existence of multiple sensors connected to the same device.

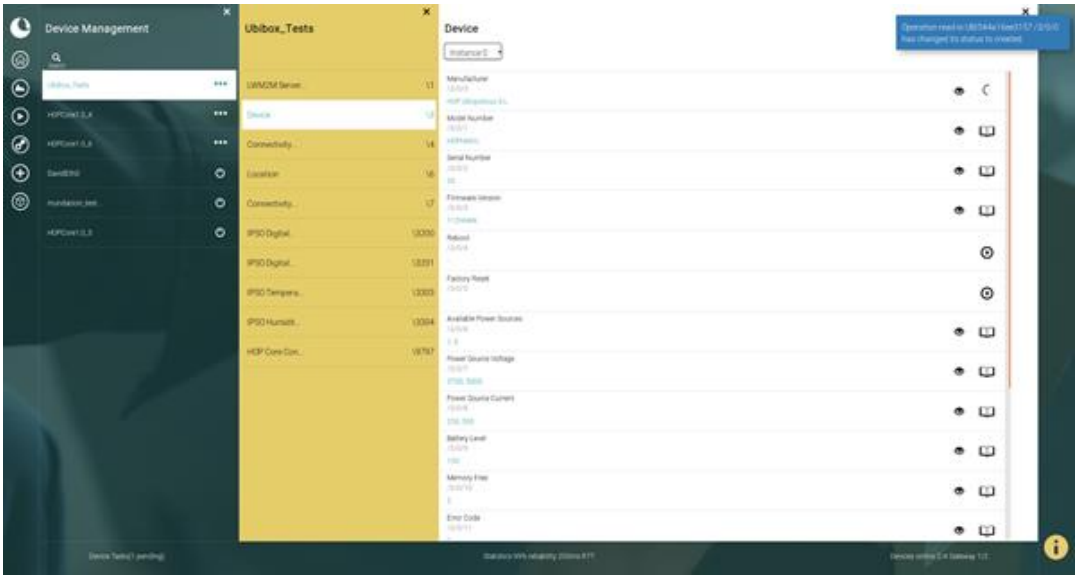

Figure 13: Homard Device Management

The Network Health Monitor service, shown in the figure below, provides an insight into the quality of the link between the device and the LwM2m server, besides other parameters that allows us to know the device load in compute and memory. This service is deployed as an external Homard module that will be notified of any LwM2m messages arriving or sending to the LwM2m server. Through this information we can store the messages exchanged, infer problems in the device and calculate interesting values of the link such as round-trip time, number of messages exchanged, number of failures against requests made or the real time that occurs between device updates.

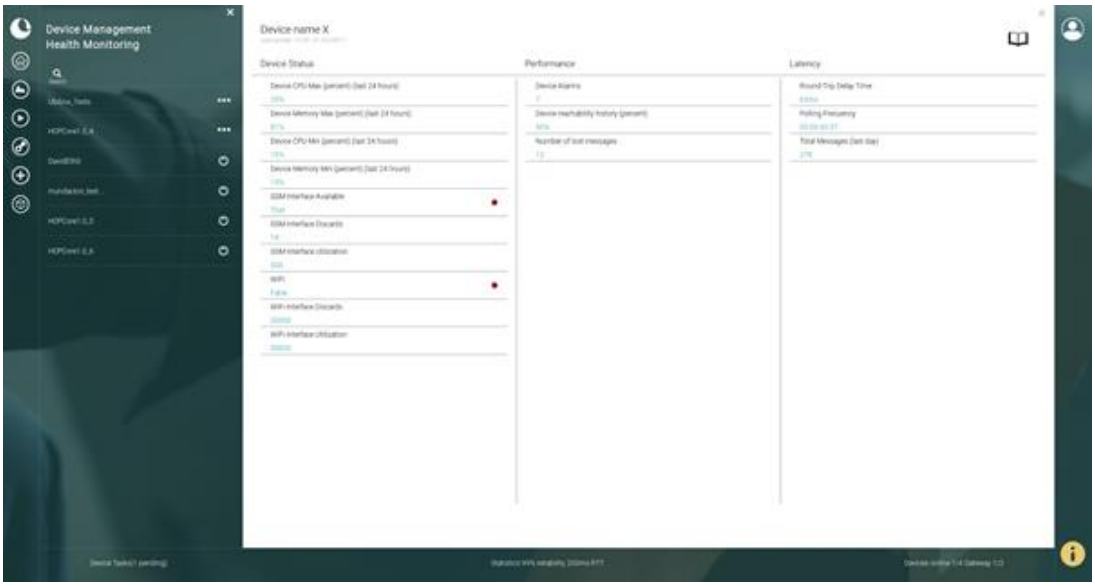

Figure 14: Network Health Monitor

Task Orchestration for CPPS offers a cloud-level component for OMA LwM2M devices. It uses other background components (i.e. Activiti for the BPM-based logic) and takes advantage of integrations such as FIWARE Orion Context Broker for enabling Field level devices using OMA LwM2M to be supported by other BEinCPPS components.

The highlighted functions are:

- Interconnection of IoT devices based on OMA LwM2M with BPMs: The Activiti BPM Engine module is adapted in order to enable its functionalities for IoT and industry processes. More other tools are supported, such as proprietary solutions integrating ESPs (i.e. SAP software), and other online tools as Fujitsu RunMyProcess.
- IoT devices management platform (Homard): Task Orchestration for CPPS offers Homard functionalities due to the fact that is built on top of it. The uses of this device management platform are defined above in this section.
- Bootstrap and deployment of the IoT services over the Cloud infrastructure: The definition of logic such as historical data storage, subscription to events, data logging, etc. needs the deployment of microservices built on containers such as OpenStack images and Dockers swarm. To make this interoperability with the IoT devices and the services, Task Orchestration for CPPS has a Bootstrap Server integrated in order to deploy an run the defined logic.
- Complementary components for specific hardware:
    - Device URL Manager Component: This component takes advantage of Industrial Physical web technology via BLE (Bluetooth Low energy). IoT devices that support this functions are called beacons,

which are designed to facilitate the users by sending Web page announcement. The motivation of this tool is to improve the interface between the machinery of a factory and the workers.

- FIWARE IoT Agent OMA LwM2M for FIWARE Orion Context Broker: as previously mentioned, a Bootstrap server integrated in order to deploy, between others, a micro-container that includes the IoT Agent for OMA LwM2M in order to carry out the mapping and translation of the data into the correspondent OMA NGSI APIs and ETSI ISG CIM data models.

Deploy environment

There are currently two Homard instances deployed: legacy and staging. Homard legacy provides the functionality aforementioned, while Homard staging provides improvements in relation to the separation of functionality in modules, as well as in the flow of the user experience through a simple and elegant user interface.

Homard legacy has been developed in Java and is deployed as a WAR. For its deployment it requires a configuration process in which a set of dependencies has to be satisfied. Dependencies as databases or other components that complement its functionality or store the service statuses.

Homard legacy implements a instance validation service to assure that every instance deployed has been verified by HOP Ubiquitous.

Homard staging shares the basic functionality of Homard legacy but focusing on the separation of its features in light and independent modules. This new architecture aims to provide a set of microservices that can be deployed and orchestrated on demand, moving from a monolithic architecture to one of distributed services in the cloud.

Homard staging is deployed and distributed through a docker service, which initializes both the server and its dependencies. In this way, a simple and coordinated deployment is achieved. This deployment is accessible on any platform with the only requirement of the correct installation of Docker.

How to gain access

From the administrative point of view, Homard is a proprietary component in which the core and key components such as data storage and BPM integration are released under Eclipse Distribution License 1.0 (BSD). Other modules as the data visualization and advanced services are registered by HOP Ubiquitous, and acceptance of their terms and conditions is necessary for their use.

From the technical point of view, Homard provides a user interface and a REST-based API through which to register and manage connected devices. For this, it will be required to have a user account that can be obtained through OAuth authentication

or directly requesting it from HOP Ubiquitous. Any information related to the use of the component as well as its modification can be found in the link provided below:

- https://homard.hopu.eu/indexPage/wiki.html