Horizon 2020 – LCE - 2017 - SGS

FLEXCoop

Democratizing energy markets through the introduction of innovative flexibility-based demand response tools and novel business and market models for energy cooperatives

D2.9 – FLEXCoop Framework Architecture including functional, technical and communication Specifications – Final Version

Due date: 30.03.2020 **Delivery Date**: 02.06.2020

Author(s): Karsten Isakovic, Peter Hasse (Fraunhofer), Hrvoje Keko, Leila Luttenberger, Jakov Krstulović Opara, Marin Bačić (KONČAR), Dimitris Panopoulos (S5), Germán Martínez (ETRa), Jordi Cipriano (CIMNE-UdL), Eloi Gabaldon (CIMNE), Katerina Valalaki (Hypertech), Dialektakos Nikos (Hypertech), Andreas Muñoz (CIRCE)

Editor: Peter Hasse (Fraunhofer)

Lead Beneficiary of Deliverable: Fraunhofer

Contributors: Etra, Hypertech, CIRCE, Koncar, S5, CIMNE, Merit

Dissemination level: Public **Nature of the Deliverable:** Report

Internal Reviewers: Katerina Valalaki (Hypertech), Peder Bacher (DTU)

FLEXCOOP KEY FACTS

Topic:	LCE-01-2016-2017 – Next generation innovative technologies enabling smart grids, storage and energy system integration with increasing share of renewables: distribution network
Type of Action:	Research and Innovation Action
Project start:	01 October 2017
Duration:	36 months from **01.10.2017** to **30.09.2020** (Article 3 GA)
Project Coordinator:	Fraunhofer
Consortium:	13 organizations from nine EU member states

FLEXCOOP CONSORTIUM PARTNERS

Fraunhofer	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.
ETRa	ETRA INVESTIGACION Y DESARROLLO SA
HYPERTECH	HYPERTECH (CHAIPERTEK) ANONYMOS VIOMICHANIKI
DTU	DANMARKS TEKNISKE UNIVERSITET
CIRCE	FUNDACION CIRCE CENTRO DE INVESTIGACION DE RECURSOS Y CONSUMOS ENERGETICOS
KONČAR	KONCAR - INZENJERING ZA ENERGETIKUI TRANSPORT DD
SUITE5	SUITE5 DATA INTELLIGENCE SOLUTIONS Limited
S5	SUITE5 DATA INTELLIGENCE SOLUTIONS Limited
CIMNE	CENTRE INTERNACIONAL DE METODES NUMERICS EN ENGINYERIA
RESCOOP.EU	RESCOOP EU ASBL
SomEnergia	SOM ENERGIA SCCL
ODE	ORGANISATIE VOOR HERNIEUWBARE ENERGIE DECENTRAAL
Escozon	ESCOZON COOPERATIE UA - affiliated or linked to ODE
MERIT	MERIT CONSULTING HOUSE SPRL

EXECUTIVE SUMMARY

This deliverable describes in detail the design of the functional components of the FLEXCoop architecture. The interfaces of each component inside the framework as well as to external communication end points are the main focus of this document.

Each component is described in a separate subsection of Section 3. These subsections are divided into five topics to provide a complete picture on how these components are designed and how they interact. In each component, the programming language as well as the libraries and frameworks used are described based on experience and problem domain. Also, the corresponding deployment strategies are covered in the component sections. The last section gives a short summary of the development since D2.6 - FLEXCoop Framework Architecture including functional, technical and communication specifications.

As this is a follow up deliverable to D2.6 this document focuses on the changes since D2.6 and should be read with D2.6 at hand.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

No table of figures entries found.

ABBREVIATIONS

AGR	Aggregator
CO	Confidential, only for members of the Consortium (including the Commission Services)
CHP	Combined heat and power
CMP	Component
CSS	Cascading Style Sheets
D	Deliverable
DER	Distributed Energy Resources
DHW	Domestic Hot Water
DoW	Description of Work
DR	Demand Response
DSO	Distribution System Operator
DSS	Dispatch Service System
DSSy	Decision Support System
MOM	Message Oriented Middleware
EV	Electric Vehicle
FLOSS	Free/Libre Open Source Software
GDM	Global Demand Manager for Aggregators
GUI	Graphical User Interface
H2020	Horizon 2020 Programme
HTML	HyperText Markup Language
HVAC	Heating, Ventilation and Air Conditioning
IPR	Intellectual Property Rights
IED	Intelligent Electronic Device
JSP	Java Server Pages
LDEM	Local Demand Manager
MGT	Management
MS	Milestone
MVC	Model View Controller
O	Other
OS	Open Source
OSB	Open Smart Box
P	Prototype

P2H	Power-to-Heat
PM	Person Month
PROS	Prosumer
PU	Public
R	Report
RES	Renewable Energy System
RTD	Research and Development
SEAC	Security Access Control
SGAM	Smart Grid Architecture Model
TOGAF	The Open Group Architecture Framework
UML	Unified Modelling Language
URL	Uniform Resource Locator
VPP	Virtual Power Plants
VTES	Virtual Thermal Energy Storage
VTN	Virtual Top Node (OpenADR)
VEN	Virtual End Node (OpenADR)
WSN	Wireless Sensor Network
WP	Work Package
Y1, Y2, Y3	Year 1, Year 2, Year 3

1. INTRODUCTION

This report is the follow up to D2.6 on the preliminary version of the "FLEXCoop Framework Architecture including functional, technical and communication Specifications" where the conceptual architecture for the FLEXCoop solution was initially presented. While the key aspects of the architecture have proven to be a good design decision in terms of functionality separation and decoupling of components, some details have changed. To fulfil the requirements regarding data protection and security, while keeping the interoperability of the FLEXCoop solution as a high priority, the inter component communication has been advanced and hardened. This includes also the introduction of new components respectively services. The details of this the Security Access Control Framework (SACF) have been described in the D4.5. The introduction of D4.5 states about this challenge:

"The interaction of the different components of the FLEXCoop architecture is a complex problem concerning security and data privacy. On the one hand, it is crucial to make sure that components can only access the data they are entitled. On the other hand, components should not need to know to which user account or physical person belongs the data it processes. By fulfilling both goals, the system can provide maximum on privacy while keeping personal data secure."

The whole communication infrastructure has been designed around OpenID 1.0 which is based on OAuth2 - both are well known open standards and are widely used in the Web domain and IoT world. As a unique feature, the FLEXCoop solution seamlessly combined OpenID and OpenADR in way that allows to keep full compatibility to existing OpenADR implementations while raising the flexibility in terms of deployment and security of the OpenADR communication in terms of missing mechanisms like key revocation and renewing.

This document follows the structure of D2.6 while not repeating its content. Therefore, the reader should have D2.6 at hand for reference.

2. CONCEPTUAL ARCHITECTURE DESIGN

This section provides an overview of the changes of the concepts and architecture decisions of this project since D2.6.

2.1. FLEXCoop architecture

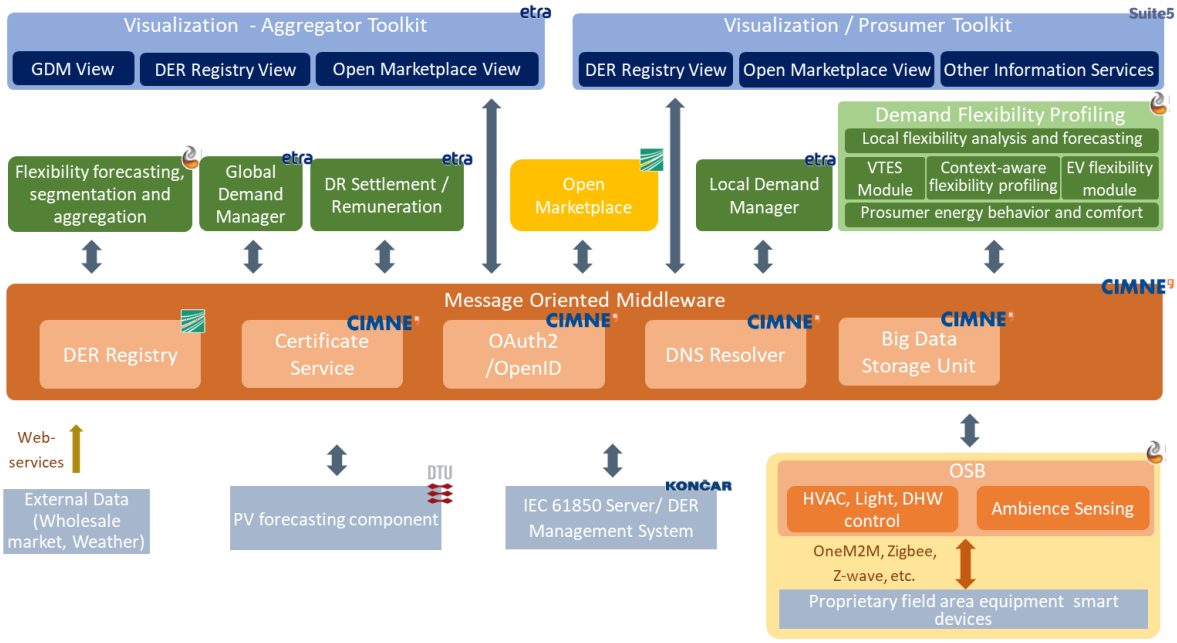

Figure 1: FLEXCoop Architecture Overview

In relation to the first version of the architecture, some changes were included:

1. The Middleware includes a central data storage, based on the big data analytics platform of CIMNE, where all the cleaned data needed for the data processing modules, is stored. The raw data transmitted by the OSBs is directly resent to the Demand Flexibility Profiling modules because it is specified that this module needs to work with online raw data. This data is then aggregated to 15 minutes frequency, stored in the middleware, and made available to the other components in real time.
2. The modules to predict the PV generation at household or district level are directly integrated into the Message Oriented Middleware as R package module.
3. USEF principles have been adopted in the FLEXCoop architecture but as the standard is a business standard and not technical one, the implementation is not strict to the USEF specifications.
4. The DER Registry has been integrated into the Message Oriented Middleware. The main reason for this decision was that the interaction between these two logical components is necessary for a lot of functionality provided by the Middleware. Combining both resulted in significant decreasing number of intercomponent messages and also speeded up the development. That said, the architecture would still allow to separate these components without deep changes if necessary.

As mentioned before three new service have been introduced which will be described in section 4.1.8.

Further differences in the diagram reflect changes in the responsibility for some of the components.

2.2. Communication Model and Data Validation

Designing and implementing a micro service architecture allows to break down complex task in less complex modules. It enables a distributed development team to work independent and can prevent delays caused by dependencies. One downside of a micro service approach, especially if different programming languages are involved is that messages between the components have to be well defined and validated to prevent incompatibilities between components. A good practise to cope with this problem is to use schema definition for all messages which define the format and possible content of a message. This way a component can verify the syntactical correctness and completeness of a message before parsing it. Developers can use the schema definition to generate interface code stubs and message objects / structs. In FLEXCoop, we use the JSON format to encode messages while the Cerberus [1] schema is used to define the format of each message. This enables developers implementing API calls to the middleware to verify their message before even talking to a middleware instance.

3. SECURITY AND AUTHORIZATION FRAMEWORK

Based on the requirements identified in D2.6 - as well as from input from other tasks - we designed an architecture which securely connects all components of the FLEXCoop solution while also providing data protection and privacy features. We based the implementation on the OAuth2 / OpenID, which is a wildly used authorization standard for the world wide web and the IoT domain. One of its key aspects is the separation of authentication and authorization. In the FLEXCoop project scope the means that personal data of the pilot users is handled by the cooperatives which need to process this data anyway. If a pilot user interacts with the FLEXCoop platform, s/he authenticates himself/herself to his cooperative which then authorizes the FLEXCoop component the user wants to use to the system. The FLEXCoop platform only gets a unique identifier and the information of the user is known to the cooperative. All data collected by the FLEXCoop solution is linked to this ID while only the cooperative can link this ID to a real person.

A user can authenticate himself to the OAuth2 server of its cooperative and authorize components of the platform to act on his mandate. E.g. the Prosumer Visualization Toolkit can only access data stored in the middleware if it has been mandated by the user to access its data. With such an authorization it can only access the data of the very user it got the token from.

This also minimizes data breaches e.g. because of security problems in a user facing component.

[1] https://docs.python-cerberus.org/en/stable/

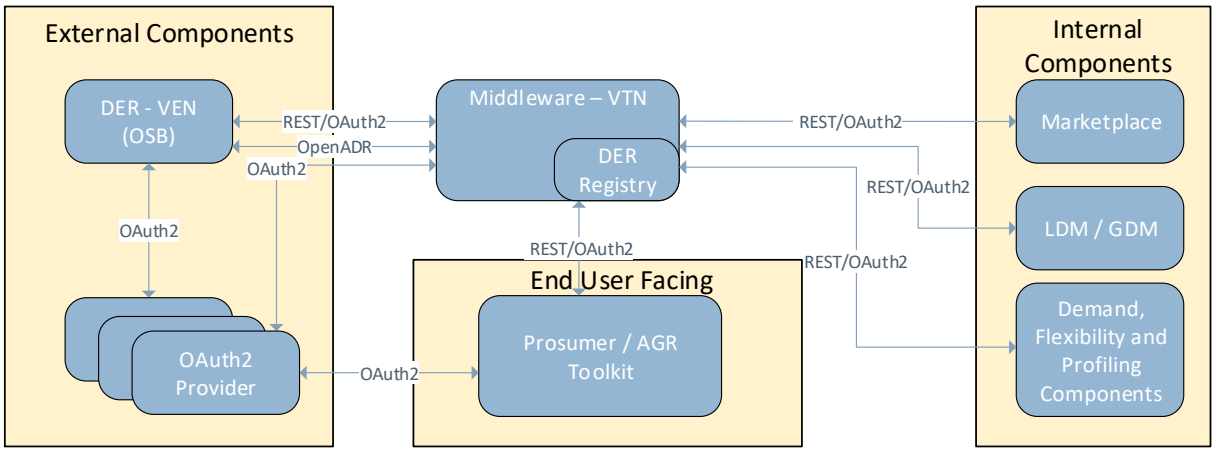

Figure 2 Communication Overview

Figure 2 Communication Overview gives a high-level overview of the component interaction in line with the developed framework. As described before components can either interact with the Middleware via REST API or OpenADR. External OAauth2 provider, operated by the cooperatives, linking their customers (the pilot users) to the platform. User facing components using the OAuth2 provider to acquire mandates for the user which are passed to through the system to authorized services provided by internal components. Internal components can either act on a user mandate or with so called worker tokens.

The evaluation of Smart Grid communication standards has resulted in the usage of OpenADR for the project, as it is a well-documented and public available standard for telecontrol of energy devices. While OpenADR itself already provides an encryption layer as well as an authorization scheme to handle access of communication nodes, we identified a shortcoming in the approach. The exchange of key material for initial personalization of end nodes like the OSB as well as key handling during the operation is not specified. To overcome this shortcoming, we developed mechanisms to combine OpenADRs keys and certificates with our authorization scheme. This enabled the OSB to request the OpenADR key material from the platform with the mandate of its owner. The OSB can pass a signing request of the certificate locally generated to the platform which then uses the internal signing service to sign the certificate and pass it back to the OSB.

4. MODULES FUNCTIONAL AND TECHNICAL SPECIFICATION

For each component, we have presented a component view diagram providing information on component and its sub-components organisation including their interfaces in D2.6. In the following the development that has been performed since will be described.

4.1.1. Visualization Aggregator Toolkit

Description of design / functionality

The Aggregator Toolkit is the friendly way the Aggregator will be able to interact with the FLEXCoop system. All its functionalities have been categorized in three different groups:

- GDEM View: It allows to perform the basic operations with the portfolio of customers: visualization of data and management of DR Campaigns.

- DER Registry View: This view enables Energy cooperatives / Aggregators to access to some of the Data provided via the Middleware, more concretely they can access to the DER registry to in order to facilitate DER discovery.
- Open Marketplace View: This view allows the Aggregators to publish their offers to attract consumers and engage them in DR services.

Description of architecture

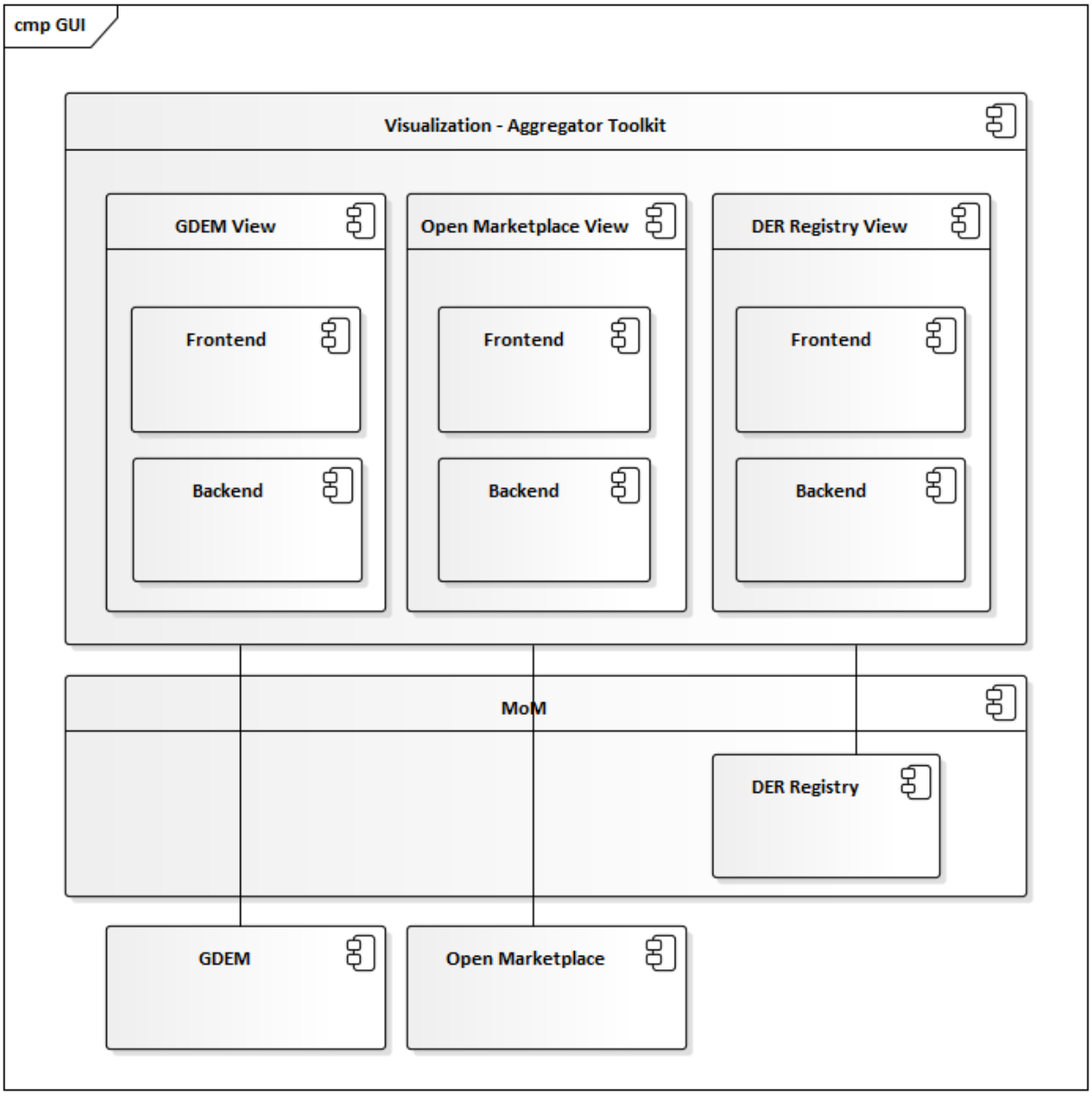

Figure 3: Visualization Aggregator Toolkit CMP diagram

Each view is split in two components:

- Frontend: Using HTML, CSS and JavaScript in the Aggregator's web browser showing the interactive GUI.

Backend: written in Node.js that serves the web pages for the frontend, and interacts with the Message Oriented Middleware for requesting the needed information.

This GUI is a Meteor application where the backend has been developed in JavaScript and the frontend uses Blaze as rendering system.

Description of component interaction

When the Aggregator opens the GUI frontend, he/she has to authenticate to the MOM through the backend. Once the authentication has been successful, the information to be displayed on the current view is requested by the backend to the MOM.

Description of deployment

The web application is deployed in a Docker container at ETRA's cloud. It can be used by opening the following URL on a web browser: https://flexcoop.etra-id.com/. It is recommended to use Chrome web browser; it can be ensured that the entire web site works as expected on this web browser.

Changes since D2.6

The main difference from the previous version is the integration of the 3 views (GDEM View, DER Registry View and Open Marketplace View) into a single one (Visualization Aggregator Toolkit), so there is no need to do 3 different implementations and then to integrate them. They can be developed as a unique component including the functionalities of the 3 submodules. Also, it has been protected the whole application with an authentication system, so only users with the proper credentials are able to access to all the information accessible through this component.

4.1.2. Visualization Prosumer Toolkit

Description of design / functionality

As specified also in the 1st version of the deliverable, the role of this software component is to act as the interface application for prosumers to enable their active participation in the energy transition of local energy communities. The main features and functionalities supported by the application are in line with the different business scenarios as examined in the project, namely: (a) efficient monitoring of real-time demand data with enriched visualization (energy demand monitoring and analytics, demand forecasts as extracted from analytics services, increasing awareness around energy market), (b) efficient monitoring of and Demand Response events triggered by Aggregators towards their active enrolment in the FLEXCoop environment & self-consumption campaigns participation (notifications of demand response events triggered by Aggregators and verification of compliance with contracts), (c) Marketplace Participation (financial and economic management of contractual agreements).The latter is a main innovation of the tool as will enable the active participation of end users/consumers in local flexibility markets following negotiation towards the establishment of contractual agreements with the different market operators (DR Aggregators in FLEXCoop).

Description of architecture

The definition of Prosumer Toolkit internal architecture is in line with the functional analysis and is presented in the 1st version of the deliverable. The MVC approach has been considered for the development of the application for the consumers/prosumers. The reason for this decision was to ensure the modularity of the application, by separating the data management from the analytics and visualization layers. The component diagram for this software module is presented (updated version of the component diagram presented in the 1st version).

Description of component interaction

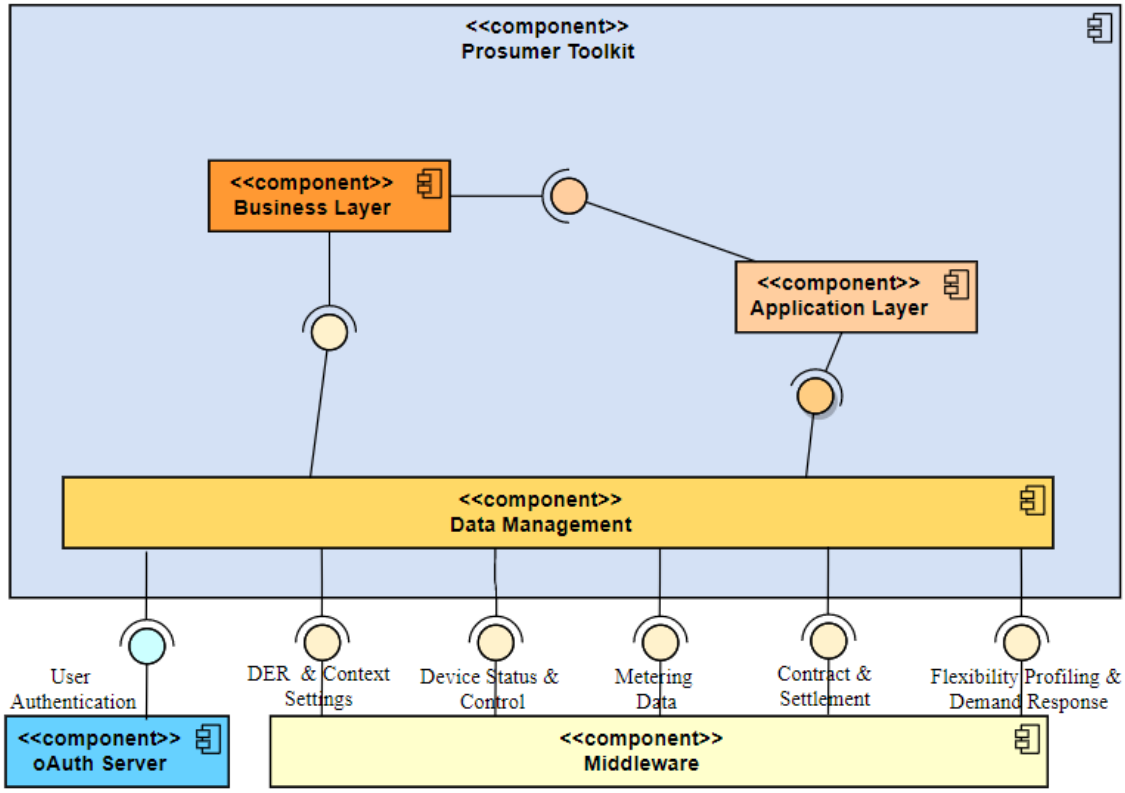

Figure 4: Visualization Prosumer Toolkit CMP diagram

Data Management Layer: The main role of this module is to define the wrappers for interfacing with FLEXCoop Middleware. While the Prosumer Toolkit will not store the dynamic logs of information required for visualization, a minimum of configuration/settings information along with business/market related parameters will be stored internally in the Prosumer application data management layer. This is actually a main update from the 1st version of the architecture. Due to the final updates on privacy/security aspects of the project, minimum information will be stored in the local database of the FLEXCoop Prosumer Toolkit.

Application layer: This is the analytics layer of the application to support the different functionalities as defined in the list of requirements:

- Enriched Visualization and Awareness: (near) real time information visualization, historical Reports visualization, DER registry information visualization

- Demand Response & self-consumption campaigns management toward the active enrolment of end users in energy market schemas as examined in the project
- Marketplace participation: contractual agreements negotiation and management to ensure the active enrolment of end users in energy markets

The definition of the different analytics is in line with the initial list of requirements.

Business layer: This is the front-end layer for rendering and presentation of information to the associated end user devices.

More details about the architecture of the FLEXCoop Prosumer Toolkit along with the details of the development process and the final views of the component are presented in the 2 versions of the FLEXCoop Prosumer Toolkit in WP6 (D6.3 & 6.7 respectively)

Description of deployment

In order to present through the Prosumer application, all the aforementioned information, interfaces with FLEXCoop Message Oriented Middleware were defined (on the basis of FLEXCoop CI M definition) with the requested information for visualization to be available from different FLEXCoop software component (through Middleware). More specifically:

- OSB & DER registry for retrieving home environment metering and sensing data
- Demand Flexibility Profiling Module for retrieving flexibility profiling related data
- Global Demand Manager; subscribing to receive DR events
- DR settlement and remuneration module requesting information about settlement through DR participation (business related parameter)
- Open Marketplace for Market participation; setting contracts and negotiation contractual agreement
- Demo Authorization Services to authorize consumers/prosumers to access personalized information
- Component Authorization Services to authorize Prosumer Portal to access information from other system components (Intercomponent communication)

As stated above, REST services to Message Oriented Middleware were considered for the integration. The detailed specifications for the aforementioned interfaces were defined in WP4 (Interfaces definition) and the implementation of the integration was performed as part of the work in WP6 and the integration task (T6.4).

Changes since D2.6

The overall architecture approach for the Visualization Prosumer Toolkit remained similar to the initial design as presented in D2.6. At a functional level, minor updates where considered to the different functionalities as envisioned in the 1st version. All these details are presented in the details of the Visualization Prosumer Toolkit in D6.3 & D6.7. From a technical viewpoint, the main differentiation is the increate of privacy/security requirements in the projects and the incorporation of additional authentication levels/services towards this direction. At the same page, the local database in Visualization Prosumer Toolkit was removed from the initial design and the final version is a lightweight version of the app where no data are locally stored.

4.1.3. Local Demand Manager

Description of design / functionality

The Local Demand Manager (LDEM) component is located at building level. Its role is to monitor in real time the context and operational conditions, and by combining information coming from the Demand Flexibility Profiling to select the optimal DR control strategies, when requested by the Global Demand Manager (which is located at district level).

Description of architecture

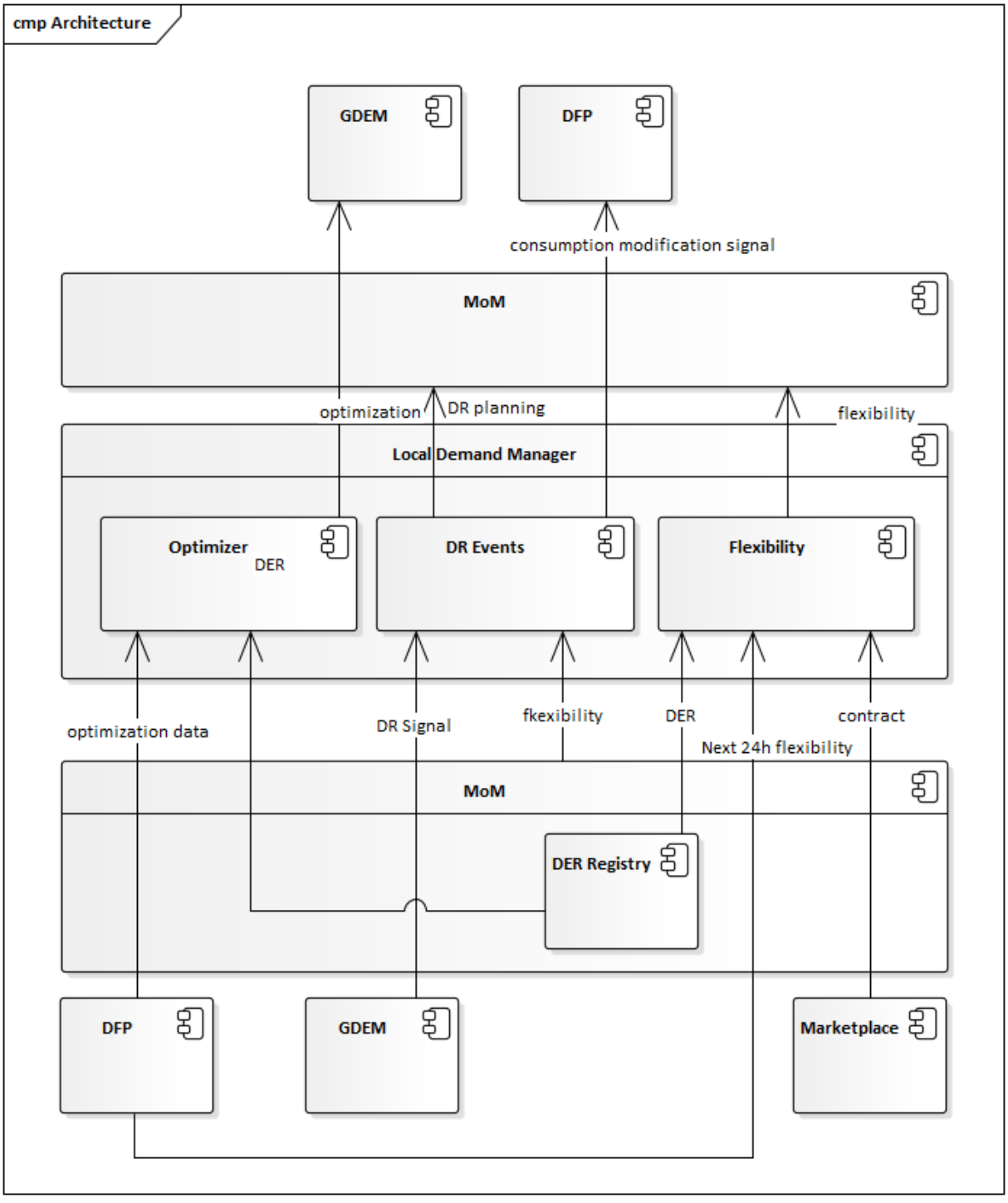

Figure 5: Local Demand Manager CMP diagram

This component contains 3 different sub-components, each one in charge of:

- Optimizer: (Only for the Spanish pilot site) Once a day it takes into account the forecasted self-production of energy of the user and creates a plan for adapting his/her consumption patterns for the next 24 hours in order to take advantage as much as possible of this generation's prediction.

- DR Events: As soon as the Global Demand Manager triggers a DR Campaign and this LDEM is informed about that, it creates a plan taking into account all its available devices and their forecasted flexibility, and sends consumption modification signals to them through the Demand Flexibility Profiling.
- Flexibility: Once a day it is gathered the next 24 hours flexibility of the devices under the control of the LDEM. That information is crosschecked with the contracts and the real flexibility for the next 24 hours is published on the Middleware.

The entire module has been developed in Node.js as a server application.

Description of component interaction

Communication between Local Demand Manager and the rest of components will be done via the Message Oriented Middleware.

- Optimizer: First of all, it gets from the DER Registry the devices under its control, and then the needed information from the DFP for being able to do the optimization. The results of this process are communicated to the GDEM.
- DR Events: As soon as the GDEM triggers a DR Event, it gets the stored flexibility information on the MOM for elaborating the plan to be followed during the DR Campaign. At each step of that plan, consumption modification signals are communicated to the DFP.
- Flexibility: Once it has got all the Devices under its control from the DER Registry the Devices under its control. Then it gets their next 24 hours flexibility forecast from the DFP and the published contracts on the Marketplace for calculating the flexibility that will be available for the next 24 hours, and it is stored on the MOM.

Description of deployment

The LDEM is deployed in a Docker container at ETRA's cloud. The LDEM provides an individual demand calculation for each user represented by one OSB each. It has to be highlighted that the implementation of the LDEM therefore provides individual "virtual LDEM" functionality for each of the users while running as one application instance.

Changes since D2.6

Comparing this version with the previous one, the main difference is the deployment of this component. Instead of having 1 deployment per dwelling, it will be only one Local Demand Manager running instance handling the information of all the users within the project.

4.1.4. Global Demand Manager

Description of design / functionality

The Global Demand Manager is continuously analysing the available demand/storage flexibility and the incoming DR signals to decide on-the-fly the optimal configuration of demand-based dynamic Virtual Power Plants (VPPs) to timely respond and provide the required flexibility to the grid.

Description of architecture

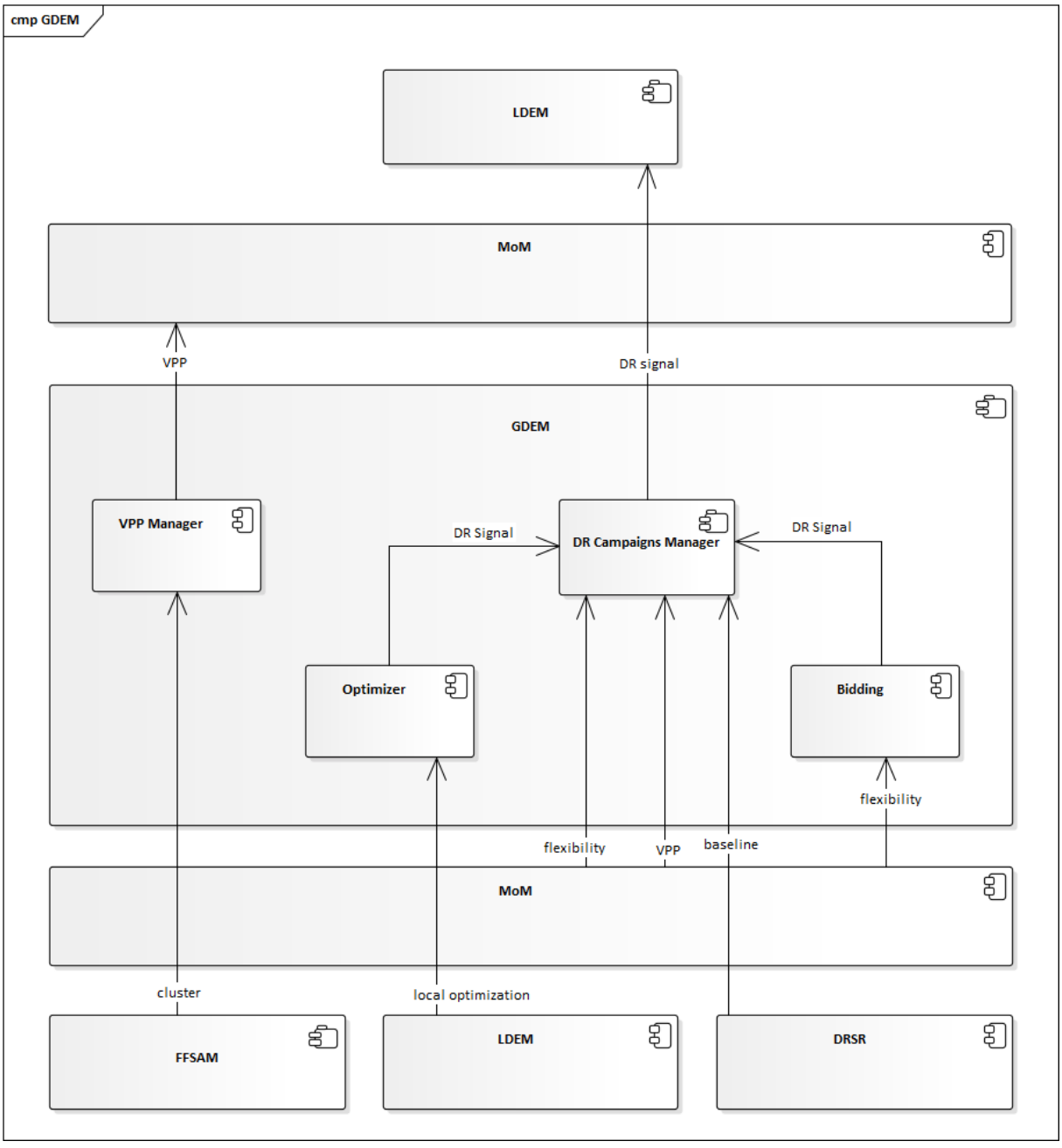

Figure 6: Global Demand Manager CMP diagram

This component contains 4 different sub-components:

- Optimizer: (Only for the Spanish pilot site) Once a day it gathers the results of all the local optimizations and performs a global optimization for calculating the amount of energy that has to be purchased on the market for the next day.
- Bidding: (Only for the Dutch pilot site) Once a day it gathers all the flexibility of the entire portfolio, taking into account the Dutch aFRR market restrictions, and places a bid on that market.

- VPP Manager: Once a day creates/updates the VPPs taking into account the results of the clustering algorithms.
- DR Campaigns manager: The output of the Optimizer and Bidding sub-components will be a new DR Campaign event. On this sub-module it will be processed that DR signal, along with some other parameters, for elaborating the planning for being able to succeed on that DR Campaign.

The entire module has been developed in Node.js as a server application.

Description of component interaction

Communication between Global Demand Manager and the rest of components will be done via the Message Oriented Middleware.

- Optimizer: It gets from the LDEM all the local optimizations that have been performed today, and when the time arrives it dispatches the proper consumption modification signals to all of them.
- Bidding: It gets from the MOM the available flexibility that has been previously calculated on the LDEM.
- VPP Manager: It gets from the FFSA the results of the different clustering algorithms.

Description of deployment

The GDEM is deployed in a Docker container at ETRA's cloud.

Changes since D2.6

The major changes of this component, comparing it with its previous version, is the inclusion of more submodules for fulfilling the requirements of the 2 business scenarios that are being implemented in the project (which are documented on D7.2 "Evaluation Framework and Respective Validation Scenarios").

4.1.5. Flexibility forecasting, segmentation and aggregation

Description of design / functionality

This component comprises an analytics platform, a tool for the aggregator. The module comprises a powerful tool for aggregators (energy cooperatives). It facilitates the management of the consumer demand and flexibility profiles enabling aggregator to forecast and decide upon the optimal breakdown of Demand Response (DR) signals to device control actions. This FLEXCoop module clusters and segments residential buildings / assets based on their actual, locally estimated demand flexibility incorporating information about building infrastructure and use by occupants. The component performs a multidimensional analysis, correlation and efficient management of prosumer profiles, flexibility and response capacity to DR signals (as described in detail in the deliverable D5.2 "FLEXCoop Flexibility Forecasting, Segmentation and Aggregation Module – Preliminary Version").

The module seeks for the most efficient Distributed Energy Resources (DERs) for each service request based on criteria defined by the aggregator through the aggregator User Interface (UI). To achieve this, the module performs portfolio segmentation based on pre-defined criteria like

the location of the building and the individual consumption of the consumers, in order to achieve the optimum dynamic and spatio-temporal segmentation of consumers' flexibility. In practice, the aggregators are offered flexibility-based portfolio clustering services, according to the service/market that they want to participate.

Description of architecture

The internal architecture of the FFSA module along with the relevant interfaces are depicted in the figure below.

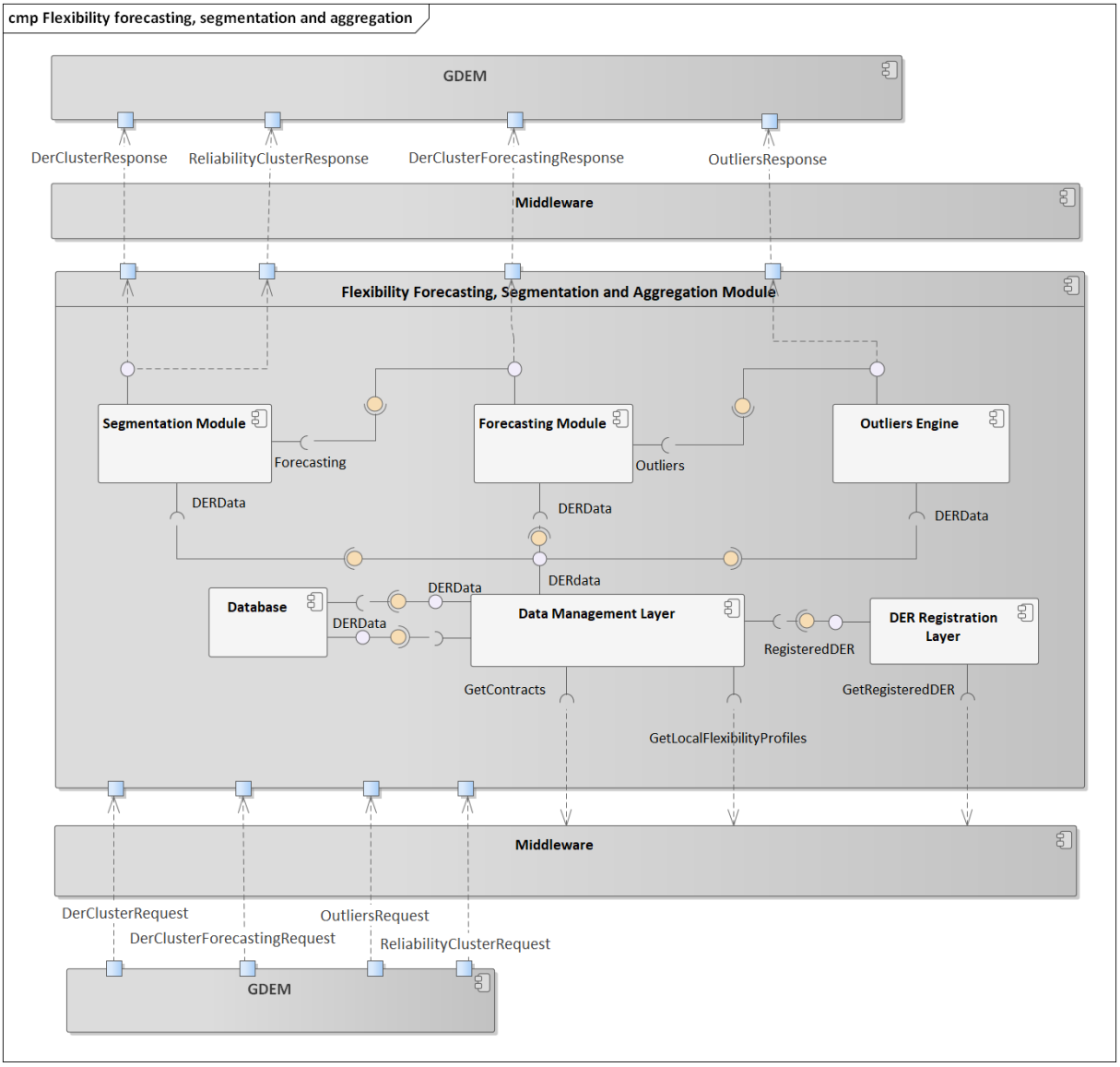

Figure 7: FFSA CMP diagram

As shown in the figure, the FFSA contains three basic submodules (see Figure 7), activated upon request by its interface with the Middleware and have the following functionalities:

- Segmentation Module: This submodule when triggered by the Aggregator in the Aggregator toolkit (through the Middleware) activates the segmentation and clustering services based on specific pre-defined criteria. The criteria that are available are:
    - Location
    - Type of device
    - Amount of flexibility
    - Customer reliability
    - Contract type and validity
    - Number of activations within an active contract
    - Self-Consumption
- Forecasting Module: This submodule, when there is a request, executes the forecasting functionality. In particular, it provides a forecast of flexibility (and a baseline if requested) for the assets that belong to specific segments as these have been defined by the segmentation module based on the criteria as defined above.
- Outliers Engine: In the same manner, this submodule is responsible for the outliers' detection. This can be for example the identification of a specific type of assets that for some reason cannot provide flexibility for a long period of time.

Description of component interaction

In the software architecture described above, RESTful services to external components have been exposed to support the functionality of the flexibility forecasting, segmentation and aggregation module. This way, the flexibility forecasting, segmentation and aggregation module is interfacing through the Message Oriented Middleware with:

- The Aggregator UI for data visualization
- The DER registry and Open Marketplace for searching registered DERs and their contractual characteristics

The Global Demand Manager (GDEM) to facilitate the real time DSS optimization and selection of prosumers to participate in DR campaigns Component Authorization Services to authorize FFSA module to access information from other system components
Furthermore, the FFSA module will request historical data on energy consumption, flexibility profiles, registered DERs and DR campaigns by the middleware whenever required.

The available already implemented endpoints are depicted in Figure 7 and include:

- DerClusterResponse as a response to a DerClusterRequest
- DerClusterForecastingResponse as a response to a DerClusterForecastingRequest
- ReliabilityClusterResponse as a response to a ReliabilityClusterRequest
- OutliersResponse as a response to an OutliersRequest

Description of deployment

As it has already been described in the D2.6 "FLEXCoop Framework Architecture including functional, technical and communication Specifications - Preliminary Version" and also in the D5.2 "FLEXCoop Flexibility Forecasting, Segmentation and Aggregation Module – Preliminary Version", the development of the core application has been based on Java and the whole framework consists a J2EE application. A MySQL database has been installed to manage model parameters, information and data retrieved from the Message Oriented Middleware.

The software architecture relies on the MVC (Model View Controller) pattern. To this end, Spring MVC framework has been used. Hibernate ORM (version 5) has been also used for mapping the object-oriented models to the relational database. Finally, the WEKA library has been used for the implementation of machine learning algorithms for data mining tasks.

The flexibility forecasting, segmentation and aggregation module is a cloud application, ensuring reliability and stability. The deployment of the flexibility forecasting, segmentation and aggregation framework is hosted in an Apache Tomcat server (open source implementation of the Java Servlet, JavaServer Pages and Java WebSocket technologies) operating as an application server.

Changes since D2.6

The overall architecture approach for the FFSA remained similar to the initial design as was presented in D2.6. Minor updates were considered to the different functionalities as envisioned in the 1st version of the FLEXCoop architecture. The respective details are provided in the corresponding deliverables, namely the D5.2 "FLEXCoop Flexibility Forecasting, Segmentation and Aggregation Module – First Version" & D5.6 "FLEXCoop Flexibility Forecasting, Segmentation and Aggregation Module – Final Version". The basic changes / enhancements are briefly presented below.

- **Functional point of view**: The criteria for segmentation and clustering were fully defined based on requirements coming from other components (mainly the Global Demand Manager) while ensuring the realization of the FLEXCoop business cases. Furthermore, the clustering functionality based on the reliability of prosumers in providing DR services was added to support the GDEM's needs.
- **Technical point of view**: the main differentiation is the incorporation of Component Authorization Services to authorize FFSA module to access information from other system components following the FLEXCoop SEAC framework requirements.

4.1.6. DR Settlement / Remuneration

Description of design / functionality

The Demand Response Settlement and Remuneration (DRSR) elaborates an objective and accurate baseline of current energy performance/consumption of prosumers, and measures the flexibility that has been delivered by them during a DR event.

Description of architecture

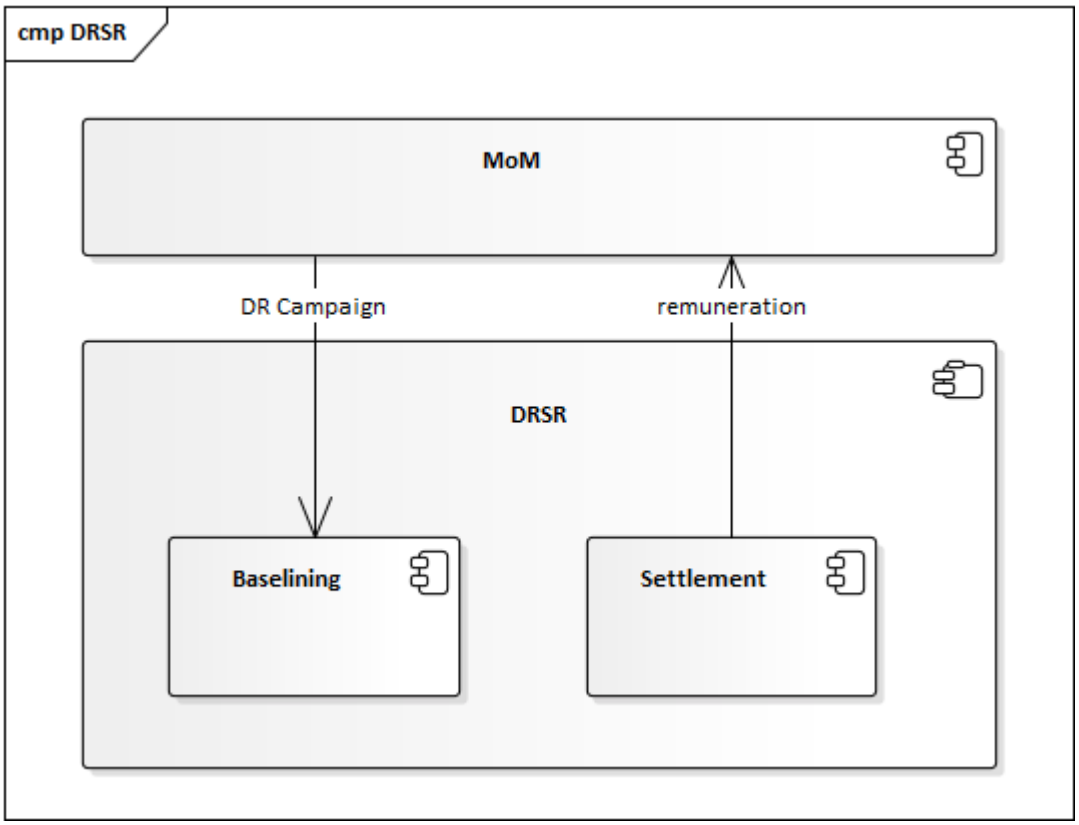

Figure 8: DR Settlement and Remuneration CMP diagram

This component contains 2 different sub-components:

- Baselining: When a DR signal has been received, firstly it has to be defined the baseline to be followed for the duration of the entire DR Campaign.
- Settlement: Once the DR Campaign has finished, the users that have participated (or/and should have done that) will be remunerated accordingly with their actuation.

Description of component interaction

The DRSR communicates only with the MOM for getting the DR Campaigns that have been received on the GDEM, and for publishing the remuneration at the end of those campaigns.

Description of deployment

The DRSR is deployed in a Docker container at ETRA's cloud.

Changes since D2.6

No particular changes have been made on this component from the previous version.

4.1.7. Demand Flexibility Profiling

Description of design / functionality

The Demand Flexibility Profiling (DFP) is the main component of the FLEXCoop architecture responsible for local flexibility profiling at asset (device) level. It is capable to construct (by utilising real time contextual values):

- personalised thermal comfort profiles;
- dynamic visual comfort profiles;
- personalised DHW demand profiles.
- EV usage profiles

Using these profiles, the DFP estimates context-aware flexibility of the specific loads (HVAC, lights, DHW) of individual prosumers as well as the flexibility that can be offered by EVs in both V2G and G2V operation modes.

Description of architecture

The internal architecture of the Demand Flexibility Profiling module along with the relevant interfaces are depicted in the figure below.

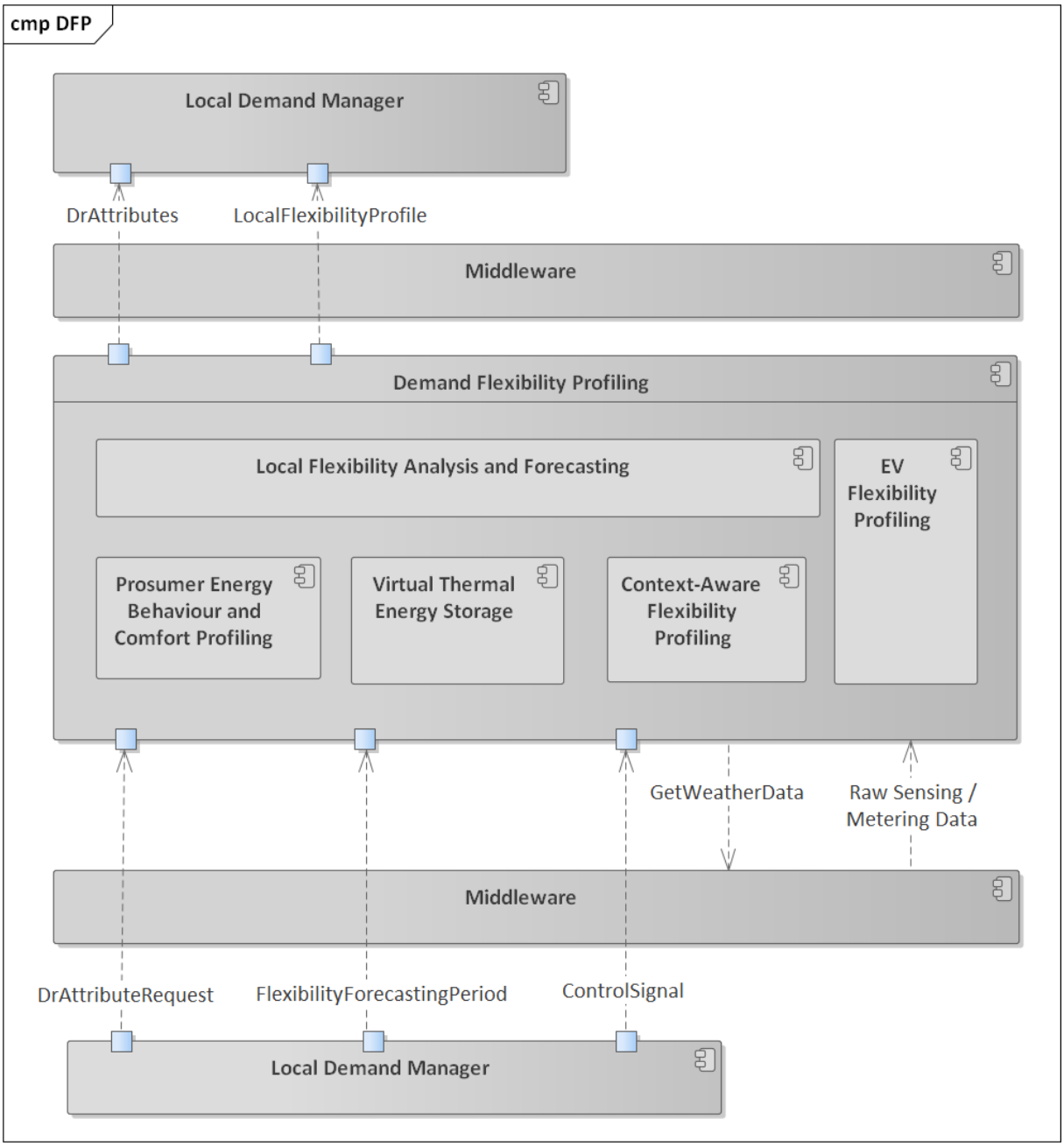

Figure 9: Demand Flexibility Profiling CMP Diagram

As shown in the figure and has already been described in the D2.6 and the D5.1 "FLEXCoop Demand Flexibility Profiling Framework" is composed from the following sub-modules:

- Prosumer energy behaviour and comfort analysis engine that identifies user's actual preferences in terms of Heating, Ventilation and Air Conditioning (HVAC), lighting and water heating loads operation, using data streams coming directly from ambient sensors and control actuators connected to OSB.
- Prosumer context-aware flexibility profiling engine that calculates and provides the potential amount of demand flexibility from controllable devices, namely HVAC and lighting devices

- Virtual Thermal Energy Storage Module (VTES) that calculates and provides the thermal storage capabilities of the prosumers' dwellings and DHW loads

Local Flexibility Analysis and Forecasting that coordinates the different flexibility layers defined above. EV flexibility profiling that calculates and provides the flexibility profiles of Electric Vehicles (EVs) charged at home. As have been described in D2.6, each of these sub-modules include:

- The required engines that perform the relevant algorithmic framework towards performing the functionalities of these sub-modules (as listed above)
- A data management layer responsible for the data retrieval and orchestration
- A database to store temporarily the data required for the algorithmic framework to be run

Description of component interaction

In the software architecture described above, RESTful services to external components have been exposed to support the functionality of the DFP module. This way, the DFP is interfacing through the Message Oriented Middleware with:

- The OSB towards receiving real time data from pilot premises and triggering control commands to DERs;

- The Local Demand Manager for providing device/ asset flexibility profiles and DR attributes as well as for receiving control signals in terms of power modification (at device / asset level)

- The prosumer portal for providing flexibility profiling related data (in real-time considering that the historical such data will only be stored and ,thus, provided by the middleware) Component Authorization Services to authorize DFP module to access information from other FLEXCoop system components

The available already implemented endpoints are depicted in Figure 9 and include:

- DrAttributes as a response to a DrAttributeRequest

- LocalFlexibilityProfile as a response to a FlexibilityForecastingPeriod

  ControlSignal

Description of deployment

The development of the Demand Flexibility Profiling component is based on Java/J2EE using a MySQL database to manage model parameters, information and data collected through the Message Oriented Middleware.

The software architecture relies on the MVC pattern providing modularity, ease of collaboration and reusability. In particular, the Spring MVC framework has been used to provide the MVC pattern as well as a number of ready components. This way, DFP has been developed as a flexible and loosely coupled component. Hibernate ORM (version 5) has been also used to map

the object-oriented models to the relational database. Finally, WEKA library is used for data mining. RESTful web services have been exposed to external components to support the functionality of the Demand Flexibility Profiling component.

The Demand Flexibility Profiling component is a cloud-based reliable and stable application responsible to perform analytics over prosumer data towards the extraction of comfort, DER, thermal storage and EV profiles. The deployment of the Demand Flexibility Profiling component is hosted in an Apache Tomcat server (open source implementation of the Java Servlet, JavaServer Pages and Java WebSocket technologies) operating as an application server. In addition, Apache server is used as a web server.

Considering the EV flexibility profiling sub-module, this has been developed in MATLAB as a separate stand-alone component because its functionality can be run without the need for cooperation with the rest of the sub-modules of the DFP. Conceptually, it can be considered part of the DFP (as shown in Figure 9) because it provides local flexibility profiles (of the EVs charged at home) but from the deployment point of view, it is a separate component built and operated at CIRCE's premises. In addition, GAMS modelling system has been used for solving the relevant optimisation problem as well as Windows Task Scheduler for bash scripting.

Changes since D2.6

The overall architecture approach for the DFP remained similar to the initial design as was presented in D2.6 and the D5.1. Minor updates were considered to the different functionalities as envisioned in the 1st version of the FLEXCoop architecture. The basic changes / enhancements are briefly presented below.

- **Functional point of view:** the main differentiations are listed below:
  - The DrAttributes API was deemed necessary to be developed and provided to other FLEXCoop components (mainly the Local Demand Manager) apart from the flexibility profiles themselves. This came as a requirement to fully cover and demonstrate the Dutch business case.
  - Data are stored only temporarily in the relevant components for the respective algorithmic framework to be supported and run. Thus, no historical data are stored in DFP for a long period of time.
- **Technical point of view**: the main differentiations are listed below:
  - the incorporation of Component Authorization Services to authorize DFP module to access information from other system components following the FLEXCoop SEAC framework requirements
  - Only RESTful web services were exposed for the interfacing of the DFP with other components. As described and explained earlier in this document, OpenADR was only supported in the OSB – Middleware communication
  - The EV flexibility profiling sub-module was developed as a separate stand-alone component as described in detail in the previous section

4.1.8. Middleware

Description of design / functionality

Although the main functionalities were already defined in the previous version of this document (D2.6), some changes and improvements have been introduced through the development and implementation of the FLEXCoop platform.

The middleware is used as the main communication and storage component in the FLEXCoop platform, as described in the data protection plan (D.8.3). It is the only component that is allowed to store permanent data. This functionality is achieved by the use of a flexible and adaptable API, where each component developer partner can contribute providing their data models following the agreed JSON schema.

Also, it will enable the communication with the OSBs installed at the Prosumer's households using the standard protocol OpenADR in order to gather all the information recorded in the user's installations and also sending them the demand response events defined by the other components (i.e. LDEM, prosumer portal).

In relation to the communication with external data provider sources required by the project is performed in the following way:

1- The outdoor temperature, the wind speed and the wind bearing data are obtained from a weather web service of the company Dark Sky[2]. These climate data are based on the approximate location of each household. Additionally, the global incident solar radiation in a planar surface is obtained from the Meteo Galicia[3].
2- The day ahead electricity prices needed for the Spanish use case are gathered from Information service of the Spanish TSO (Red Eléctrica Española). This information service is called e.sios[4] and it habilitates a public API to periodically gather these data.

Finally, it will also incorporate the FLEXCoop solution's Security and Access Authorization Framework (SEAC) components to ensure security and privacy when using the external services installed on the aggregator's side.

Description of architecture

The final solution for the Middleware is divided in the following components: (i) an OpenADR communication interface to communicate with the OSBs installed at the prosumers' households. (ii) a flexible and easy-customizable RESTFul API to enable the communication among all the FLEXCoop components using the json schema message definitions. (iii) an OpenID authentication service, to allow the worker authorization of components in the whole system. (iv) a certificate authority (CA) to provide certificates to all the OSBs and enable component customization A distributed data base is used for massive data treatment and storage of large volumes of data, based on CIMNE's big data analytics platform (ENMA)[5]. ENMA is

[2] https://darksky.net/poweredby

[3] https://www.meteogalicia.gal/

[4] https://api.esios.ree.es/

[5] https://github.com/BeeGroup-cimne/ENMA

composed by (v) a data storage layer and (vi) a Big Data Analytics Cluster to execute some analytics, data cleaning and modelling.

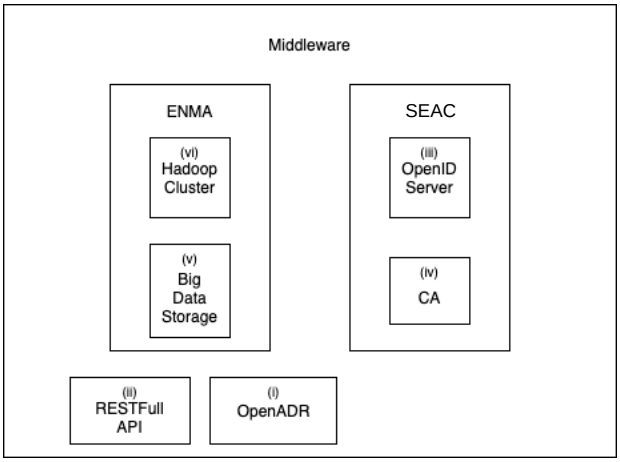

Figure 10: Middleware Components diagram

Description of component interaction

The Middleware component interacts with all the FLEXCoop components in order to allow the communication among them. All the other components have implemented the corresponding API endpoints and models required to allow the communication.

Description of deployment

The Middleware is deployed in a cloud-based server, accessible for all the other components through HTTPS and secured using the SEAC.

The development of the Middleware component is based on Python 3, providing an OpenADR 2.0b implementation using Flask, a Python Eve general purpose RESTFull API and the Django applications for the OpenID and CA servers.

The BigData Storage and Data Analysis implementation based on ENMA is implemented using Apache Hadoop, providing a secure and reliable distributed long-term data storage, and allowing the execution of Data Analysis using MapReduce technology. Beside this, to provide fast access to the data, a short-term data storage and cache is implemented using MongoDB

Changes since D2.6

Since the D2.6, some of the components have been better defined or changed, also it was detected the lack of some functionalities which have been introduced as new components into the middleware. In this section, a summary of the changes has been compelled.

New Certificate Authorization component:

A Certificate Authorization (CA) is required in order to be able to customize and provide authorization to the OSB data, the CA functionalities are described in the SEAC deliverable

(D4.5). It provides a simple REST service which can be used with an authorization grant or access token, obtained using the aggregator's OpenID servers, to receive an OpenADR x.509 certificate. The server also contains a Certificate Revocation List (CRL), to be able to revoke certificate if it is in the needs of the project.

Components communication changes:

In D2.6, all the FLEXCoop components were defined to communicate using OpenADR between them. This was changed to the RESTFul API, thus limiting the use of the standard OpenADR to communicate the middleware with the OSBs. This change was made after realizing that the only component that can be massively changed and installed in different locations is the OSB. Thus, simplifying the communication protocol in-between, the rest of the components.

Changing OpenAM for OpenID:

When defining the Data Protection Plan (D.8.3), it was detected that the OpenID protocol was more convenient for the project. It allows the authentication and authorization of different components using an external service installed in the cooperatives side, thus making it more feasible for the project.

4.1.9. DER Registry

Description of design / functionality

The **DER Registry** provides data for DER discovery and VPP formulation and allow for successful provisioning and acquisition of specific and dedicated services from DERs.

The **DER Registry** is implemented as part of the **Middleware** and acts as a database, which hold the state of all DERs connected to the FLEXCoop solution. Fraunhofer FOKUS is the responsible partner of its implementation. DERs register on the registry via the Middleware OpenADR interface when they connect to the system and update their status in the registry on any changes. The registry needs to keep track of all DERs, which includes actively monitoring keep alive messages from the DERs and mark timed out DERs as such. For accounting and monitoring reasons, the registry keeps a log of all events passed through the registry.

Description of architecture

The DER Registry is part of the Middleware and provides is functionality via the REST API as well as the OpenADR interface.

The **DER Registry** is implemented with Python3 and the Python EVE Framework.

The DER Registry uses the MongoDB Database of the **Middleware** to store the following information:

- Information on the state of the currently registered DERs to provide this information to other components.
- Details of all DERs known to the registry

- Information on the state of the currently registered DERs to provide this information to other components
- The history of events monitored by the registry

Description of component interaction

The DER Registry interacts with the following Components:

The **Middleware** API is used to communicate with the DERs via OpenADR.

Description of deployment

As the DER Registry is part of the Middleware see 4.1.8 for deployment details.

Changes since D2.6

The DER Registry has been integrated with the Middleware in favour of the initial architecture where it was designed as a standalone component. While the architecture still would allow a stand-alone Registry, this would increase the number of messages exchanged between components drastically. Implemented as part of the Middleware the registry is part of the API of the Middleware.

4.1.10. Open Smart Box

Description of design / functionality

FLEXCoop Open Smart Box (OSB) is a smart home gateway/ device. The OSB (integrating with other components of the FLEXCoop architecture through the middleware) is the central element that facilitates the deployment of human-centric DR optimization strategies. It enables personalized control functions and automation, in a non-intrusive manner and without compromising prosumers' comfort, daily operations/ schedules for the provision of the required amounts of flexibility to energy cooperatives/ aggregators.
It consists of a modular communication and sensing/ control smart system with a threefold role:

- integration of a wide range of sensors, such as luminance, occupancy, temperature, humidity, air quality
- setting of the control interfaces to monitor/control the operation of specific devices, namely lights, water heaters and HVAC systems and
- setting of the interfaces to measure the electricity consumption of these specific devices and/ or the consumption of the whole dwelling.

The OSB is used for sensing and control while enabling communication/gateway functions for interoperability with all major Smart Home protocols and devices (e.g. ZigBee, Bluetooth, z-wave, etc.) translating it them to OpenADR compliant services. OSB performs optimized energy management at the house level through its interfacing with the Local Demand Manager and the demand flexibility engine providing human-centric intelligence functions.

OSB includes a hardware and software implementation summarized below (details information can be found in the D4.1 "FLEXCoop OSB Prototype Design", D4.2 "FLEXCoop OSB Prototype - Preliminary Version" and D4.6 "D4.6 "FLEXCoop OSB Prototype - Final Version" (to be delivered at M32):

- Hardware implementation that includes:

- the OSB hardware design
- the off-the-self sensors/ actuators/ metering that comprise the FLEXCoop Home Area Network (HAN)

OSB software implementation as presented in the following sections

Description of architecture

The internal architecture of the OSB software implementation along with the relevant interfaces are depicted in the figure below.

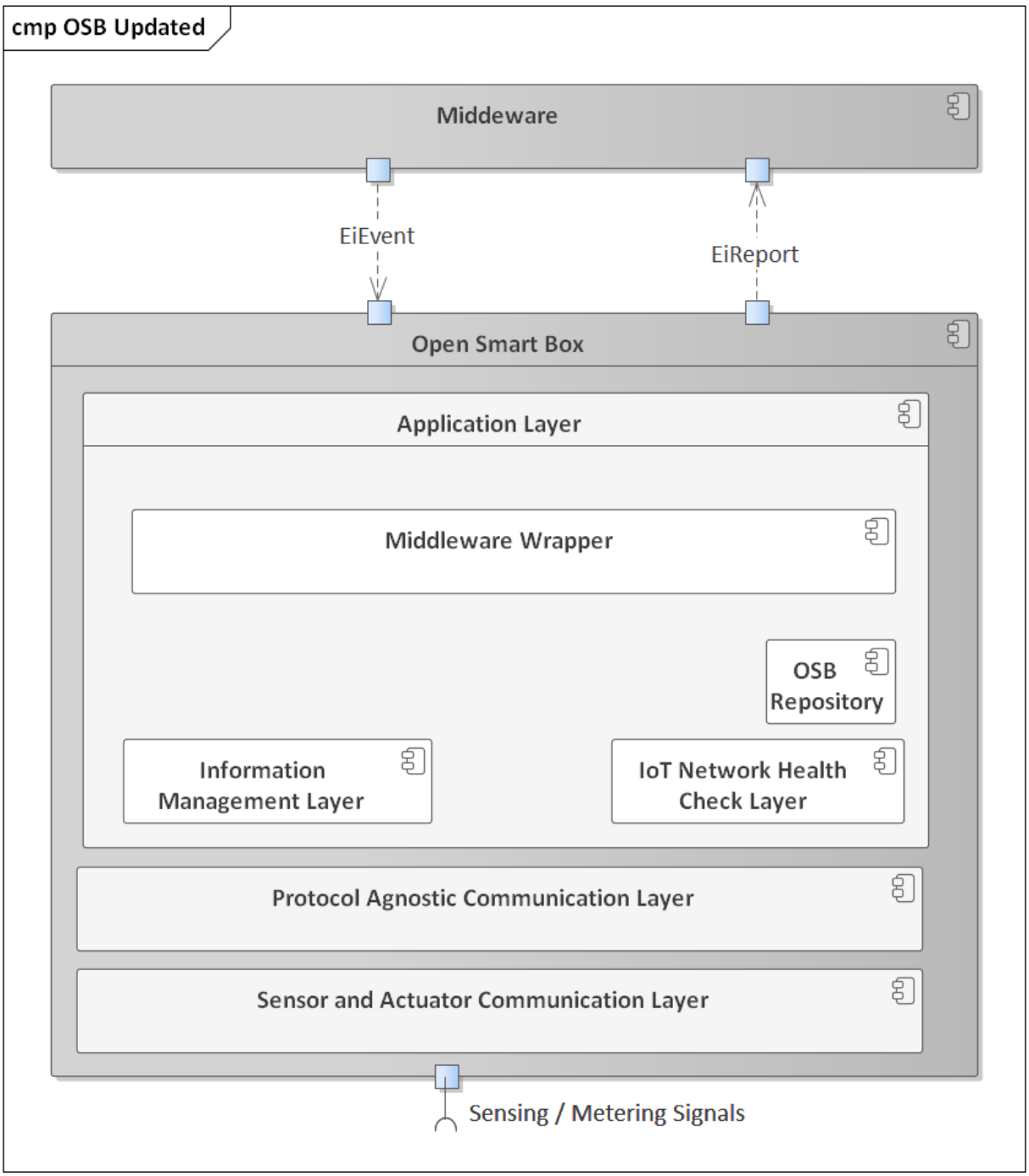

Figure 11: OSB software CMP diagram

As shown in the figure and has already been described in the D2.6 and the D4.1 and D4.2, OSB software is composed from the following sub-modules:

- The **Sensor and Actuator Communication Layer** responsible for the intercommunication with smart devices (sensors and actuators) that are installed in the home environment and are controlled by the OSB
- The **Agnostic Communication Layer** responsible for the integration of all the available functionality provided by the connected devices through different protocols in order to be collected and converted into a generic API
- The **Application Layer** that includes the basic functionality of the OSB including device discovery and commissioning through the commissioning app, control dispatching, interchanging information with the middleware using OpenADR standard, Wireless Sensor Network (WSN) monitoring functionality (Network Alerting/Recovery/Healing Features).

Description of component interaction

To interchange data with Middleware, the application layer of the OSB pushes (reports) data to the middleware using HTTPS post requests for communication using the OpenADR standard. In particular, OSB provides EiEvent and EiReport services to the middleware as depicted in Figure 11).

This way, the OSB interfaces through the middleware with:

- The Demand Flexibility Profiling engine for providing real-time sensing and energy consumption data
- The DER registry for registration of the DER devices connected to the OSB
- The Local Demand Manager for receiving signals for controlling connected devices as well as for providing real-time context – environmental conditions
- The Global Demand Manager for providing real-time context – environmental conditions and demand flexibility status
- The Prosumer portal for providing real-time sensing and energy consumption data, RES generation and storage information, etc. and for receiving settings for connected devices (e.g. comfort settings)

Description of deployment

The OSB is an in-premises local agent ensuring reliability and stability. Each OSB has been delivered with an OpenVPN setup to facilitate the maintenance/monitoring phase.

The Sensor and Actuator Communication Layer and the Agnostic Communication layer have been based on OpenHAB automation software. In particular, Java has been used to extend OpenHab in order to support the communication of OSB with all the devices needed to be installed in the pilot users' dwellings for their participation in the FLEXCoop DR framework (e.g. OSB communication with the IntesisBox®– see more information in the D4.1).

The commissioning app (included in the OSB application layer) has been built using Node.js which is based on JavaScript framework. Jade has also been used to generate all the HTML components of the commissioning app. This application was deemed necessary in order to facilitate and provide

a user-friendly approach to the commissioner to easily install and commission the OSB and then detect and commission the different smart devices that were placed in the pilot users' dwellings. The whole documentation of the OSB commissioning app including a comprehensive step-by-step user manual can be found in the D4.2.

Changes since D2.6

The overall architecture approach for the OSB software implementation remained similar to the initial design as was presented in D2.6 and the D4.1. Minor updates were considered to the different functionalities as envisioned in the 1st version of the FLEXCoop architecture. Some of them have been already detailed in the D4.2 while the final version of the software implementation will be fully described in the D4.6.

The basic changes / enhancements are briefly presented below:

- **Functional point of view:**
    - commissioning app was developed and included in the OSB application layer providing device discovery and commissioning in a user-friendly way
    - Wireless Sensor Network (WSN) monitoring functionality (Network Alerting/Recovery/Healing Features) was deemed necessary to be included in the OSB application layer for monitoring the WSN in users' dwellings
- **Technical point of view:**
    - the incorporation of Component Authorization Services to authorize OSB to access information from other system components following the FLEXCoop SEAC framework requirements
    - Only OpenADR compliant services were exposed for the interfacing of the OSB with the middleware.

4.1.11. Open Market Place

Description of design / functionality

The Open Market Place is implemented as an internal component of the FLEXCoop solution. While not providing a user facing frontend itself, the data and services provided by it are exposed via the Prosumer and Aggregator toolkits. Each contract that is handled by the FLEXCoop solution will pass the Marketplace each time it is altered, which includes acceptance of contracts as well es changes of its content. This way it is ensured that attributes of a contract can only be changed during the negotiation phase and that only syntactical correct contracts will be processed.

Description of architecture

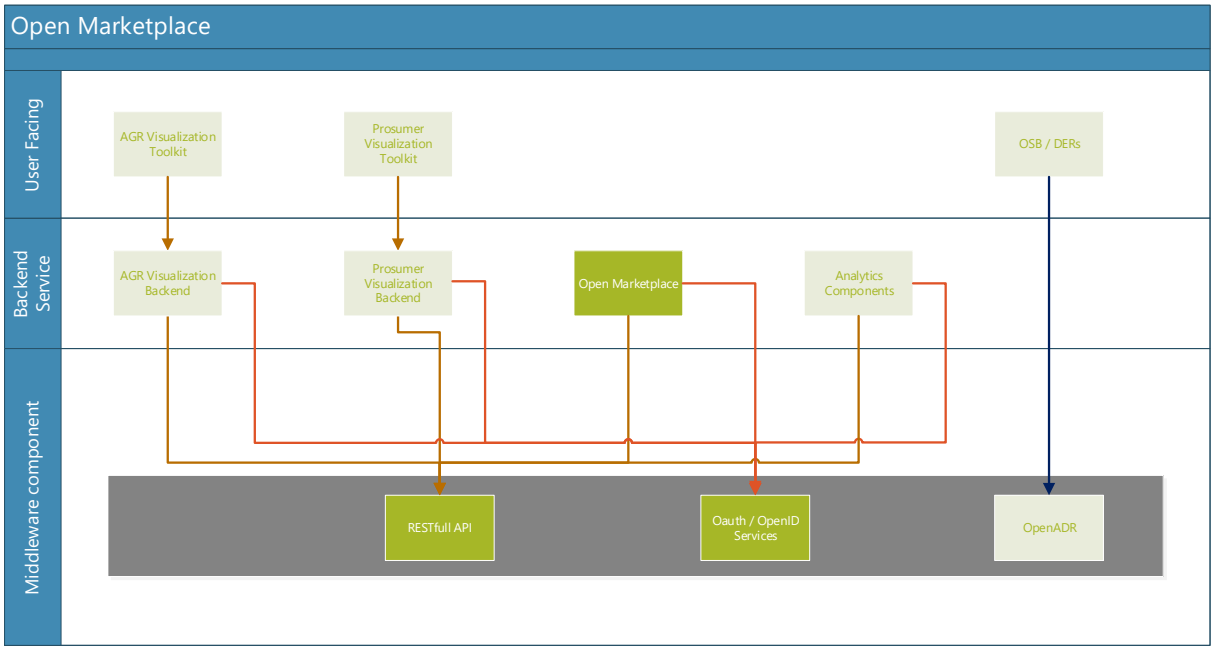

Figure 12: Open Marketplace integration

The implementation of the Open Marketplace is done using python3 and the python eve framework. The Open Market Place exposes a REST API and implements a REST client to interact with the middleware. It uses the validation functionality provided by Cerberus, which is part of eve, to check the message layout. Further logical and workflow-based rules are implemented and executed via pre- and post-hooks provided by eve. Figure 12: Open Marketplace integration shows how the Open Market Place is embedded into the FLEXCoop solution.

Description of component interaction

As stated before, the Open Marketplace interacts with the other components via the Middleware through REST APIs. While this is simple on a higher level the process from offering a contract to close it requires a lot of messages and logical steps and verification. The process of the aggregator submitting a new contract to the platform is illustrated in Figure 13: Contract submission.

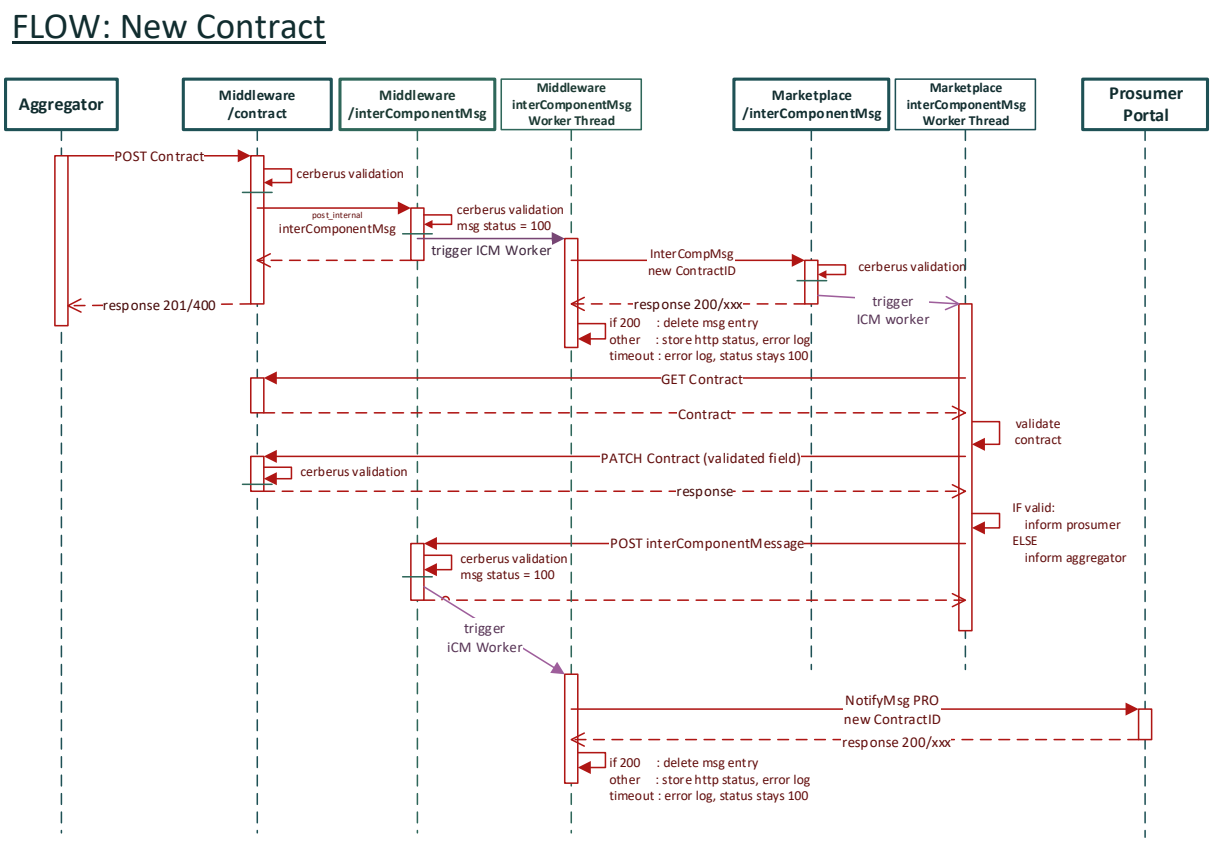

Figure 13: Contract submission

Negotiations about the terms and conditions between the Prosumer and Aggregator are supervised by the marketplace following a similar message flow.

Description of deployment

The Marketplace is pure python implementation and can therefore deployed in numerus ways. The development instance is hosted at FOKUS premises as a source-to-image container running on an OpenShift kubernetes cluster.

Changes since D2.6

- Framework was changed from flask to eve
- In memory DB was dropped
- OAuth2/OpenID is used for authorization
- Communication with the middleware is implemented RESTful in favor of OpenADR

4.1.12. IEC 61850 Server/ DER Management System

Description of design / functionality

The IEC 61850 Server / DER Management System performs the role of gateway of the FLEXCoop framework to the classic SCADA telecontrol systems. The basic FLEXCoop

implementation relies on the IEC 61850 XMPP based stack as a result from the OS4ES FP7 project.

For the situations where this support is not enough, the DER Gateway also can embed an instance of KONČAR automation PROZA platform which fully supports a number of other legacy and standardized telecontrol protocols. This requires a license of the KONČAR PROZA platform to fully function. In that case the DER Gateway is implemented in Python, interacting through the Controller functionality of PROZA platform with the automated system.

In both cases, functionally, the FLEXCoop DER Gateway component is quite analogous to the OSB, with the only difference being that the DER Gateway communicates with larger-scale devices. While the OSB communicates with in-house equipment to be actuated upon, the DER gateway issues commands and read measurements from the DER devices, using classic telecontrol protocols such as IEC 61850 locally at the DER site.

Description of architecture

The DER Management System consists of three principal components:

- FLEXCoop IEC 61850 Security-Enabled Gateway
- Local IEC 61850 communication client
- Local IEC 61850 communication server

The gateway component is a key development delivered in FLEXCoop. The novel development is that it faces the rest of the FLEXCoop framework and functions as a telecontrol gateway bridging the direct telecontrol to FLEXCoop OpenADR-based communication. This gateway fully supports the FLEXCoop communication schema for OpenADR payloads, and supports Oauth2 token-based authentication. It is implemented in Python language.

When used standalone, it only supports IEC 61850 protocol. If other protocols are required, then it runs within the Controller functionality of the KONČAR PROZA automation platform.

Description of component interaction

Seeing from the viewpoint of FLEXCoop middleware, the DER gateway operates in a very similar fashion to the OSB. It operates as an end-user OpenADR device, and performs a one-time token-based registration with the OAuth provider and the OpenADR certificate authority. The obtained certificate is then used for all further communication with the FLEXCoop middleware. The gateway then functions as an OpenADR VEN, again similarly to the OSB, and translates the JSON OpenADR message payload into the protocols available locally at the DER site. Analogously, it posts the measurements read from the DER. The DER Gateway fully supports the FLEXCoop cryptographical extension of OpenADR.

Description of deployment

All the “glue” code for the DER gateway is implemented in Python. The simplified version relies on the open-source IEC 61850 libraries developed by KONČAR within the OS4ES FP7 project. The more advanced version relies on an instance of the KONČAR PROZA automation platform. This underlying KONČAR automation platform components are cross-platform and

can be deployed on Windows and Linux operating systems, as well as in a container, and run the same Python codes to operationally connect with other FLEXCoop components.

However, the local component facing functionality of DER gateway is closely tied to the existing DER systems that have to react to the received telecontrol commands, the actual deployment configuration has to ensure proper communication channels with the existing system – and this is done manually by configuring the "local" side of the gateway.

This includes communication channels. For instance, the TCP/IP based protocols, the traffic has to reach the local automation system, and configuring the networks properly is a part of the SCADA engineering and deployment.

As stated in the D2.6, this component is designed primarily to be deployed in cooperation with the existing SCADA system. Then the DER gateway as an IEC 61850 client communicates with local IEC 61850 servers, which are often instances of a local small-scale SCADAs. Both sides of the 61850 protocol have to be configured manually: the local SCADA and the DER Gateway.

As the KONČAR automation platform offers full SCADA functionalities, too, the DER Gateway with a full license for KONČAR PROZA NET SCADA platform can also be deployed as a replacement of the existing SCADA which is then decommissioned. In that case, the DER gateway takes over the functions of local SCADA and communicates and issues commands to the existing equipment directly, using Modbus, OPC, IEC 60870-5-104, DNP3 or other telecontrol protocols. This is, however, a much more invasive task, requires engineering, functional and acceptance testing, and is expected only when existing SCADA system is already unsatisfactory.

Changes since D2.6

Detailed specification of components, detailed full support for OAuth2 token-based component authentication with the FLEXCoop middleware. Providing an option to fully replace the local SCADA with a KONČAR SCADA instance, where applicable and necessary.

5. DEPLOYMENT

As described above, the prerequisite for the software deployment is the existence of, either physical or virtual Linux or Windows machine that is able to reach the controllable assets to read the measurements and set the setpoints. Thus, the prerequisite to software deployment are: established communication channels to the controlled equipment (downstream) and to the FLEXCoop middleware (upstream), and then a manual reconfiguration of the local control system (local SCADA instance) to listen to the gateway.

Where required or where the local SCADA does not exist - the additional step is the full KONČAR SCADA engine configuration. Either way, these prerequisites have to be configured manually on a case-by-case basis.

The final DER Gateway software deployment then consists of deploying and activating a corresponding YAML configuration of the gateway, then starting the gateway. The gateway runs as a Windows service on a Linux machine or as a Linux system service on Linux machines, starting automatically upon the machine restart.

The final step is registering the gateway with the FLEXCoop system through the provided local Web-based interface (this step is equivalent to registering the OSB).

6. CONCLUSION

The FLEXCoop architecture has evolved and was extended in terms of security, privacy and platform integration. With OAuth2 as authorization framework and TLS as transport security the solution is based on well-known open standards. The integration of OAuth2 with OpenADR raises the security level of OpenADR and allows secure provisioning of end devices. By separating personal information from the data, the platform implements the least-knowledge principle and provide strong data protection already on API level. All APIs are defined by schema definition to allow fast integration with new or third-party components.