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

WP4 – DATA ACQUISITION, MANAGEMENT AND SECURITY

D4.4 – FLEXCoop Middleware and Big Data Management Platform - Preliminary Version

Due date: 31.07.2019	**Delivery Date**: 07.08.2019
Author(s): Eloi Gabaldon (CIMNE); Jordi Cipriano (CIMNE)	
Editor: Eloi Gabaldon (Editor)	
Lead Beneficiary of Deliverable: CIMNE	
Contributors: Fraunhofer, Hypertech, Grindrop, Koncar, Suite5	
Dissemination level: Public	**Nature of the Deliverable:** Demonstrator
Internal Reviewers: Laura Morcillo (ETRa), Germán Martínez (ETRa), Andreas Muñoz (CIRCE), Juan Aranda (CIRCE)	

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
GRINDROP	GRINDROP LIMITED
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

Disclaimer: FLEXCoop is a project co-funded by the European Commission under the Horizon 2020 – LCE-2017 SGS under Grant Agreement No. 773909.

EXECUTIVE SUMMARY

The current deliverable is the outcome of the first version of Task T4.3 “Middleware Configuration and Big Data Analytics Platform”. This is a demonstrator deliverable that provides the implementation of the Middleware component.

The component consists on a high performance communication Application Programming Interface (API), to interconnect all the FLEXCoop components in a secure way, an implementation of the standard OpenADR to communicate with the FLEXCoop Open Smart Box (OSB), and a massive and fault tolerant storage database and big data analytics infrastructure, to store all the required data and perform data analytics processes and models in a fast and parallelized way.

The Middleware is intended to act as a central database storage, where all the required data from the FLEXCoop components will be stored, ensuring always the conformance of the security protocol defined as an outcome of the FLEXCoop Deliverable D4.5: "FLEXCoop Security Access Control Framework (SEAC) – Preliminary Version.".

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
BSD	Berkeley Software Distribution
D	Deliverable
DOW	Description Of Work
DB	Data Base
DR	Demand Response
ETL	Extract, Transform and Load
EUPL	European Union Public Licence
FIFO	First In First Out
HDFS	Hadoop Distributed File System
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OSB	Open Smart Box
REST	Representational State Transfer
RSA	Rivest Shamir & Adleman
SOA	Service-Oriented Architecture
SSPL	Server Side Public License
VEN	Virtual End Node
VTN	Virtual Top Node
XML	eXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol
YARN	Yet Another Resource Negotiator

1. INTRODUCTION

The FLEXCoop Middleware component is designed to be the central communication and permanent data storage for all the FLEXCoop components. It aims at interconnecting all the different FLEXCoop components and allowing the communication between them. The Middleware is hosted in a dedicated server managed by the FLEXCoop partner CIMNE. Python programming language constitutes the software programming framework. The Middleware is formed by the following modules:

- OpenADR interface
- OAuth interface (linked to the deliverable D.4.5)
- API-REST interface
- Big Data analytics engine

The first version of all the modules has been made available to the consortium through the consortium software repository managed by Fraunhofer FOKUS. In Section 8 of this document, the corresponding links are provided. This deliverable is considered as a demonstrator, therefore, the main delivered products are the software codes themselves, supported by the corresponding documentation, which are all available at the consortium repository. This document is a descriptive document, which aims at providing more details on the main features and functionalities of each software module.

This deliverable is closely linked to the FLEXCoop security Access Control Framework (D.4.5) as it implements the client part of the authentication to ensure all components follow the protocol described in D.4.5 on the FLEXCoop Security Access Control (SEAC) Framework – Preliminary Version.

2. THE MIDDLEWARE IN THE FLEXCOOP ARCHITECTURE.

The FLEXCoop Middleware is the central communication component in the FLEXCoop platform. Based on the Architecture (see D2.6), the Middleware is indicated with the red frame in Figure 1.

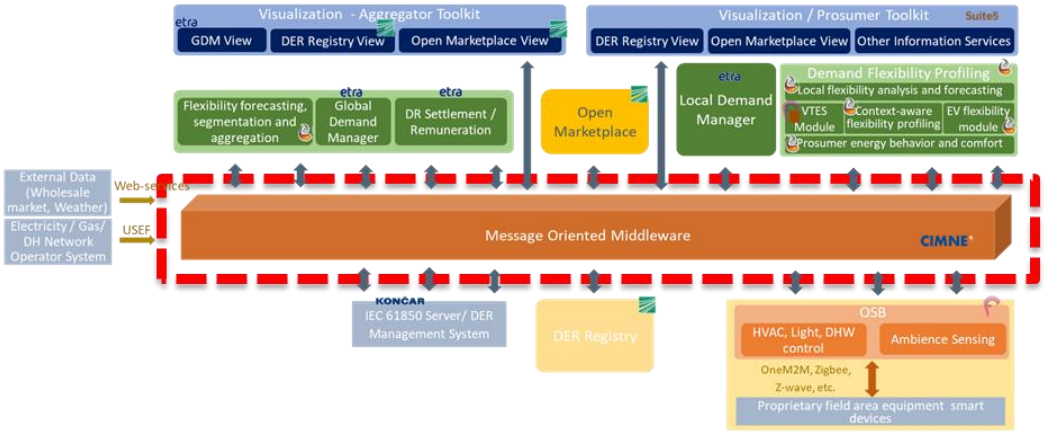

Figure 1: FLEXCoop Architecture

As it can be seen in the picture, the Middleware has communication interfaces with practically all the FLEXCoop components. This is achieved thanks to an easy configurable RESTFul API that allows each component's responsible to define their resources agreed on the FLEXCoop Common Information Model (D4.3).

3. VERSION OF THE SOFTWARE DEMONSTRATOR

This is the first version of the Middleware components based on the FLEXCoop Description of Work (DoW). However, as the Middleware becomes a central part in the integration of all the FLEXCoop components, it needs to be adapted and modified in parallel with the definitive versions of the other components. A final version of the Middleware is due to be submitted by the FLEXCoop project Month 32.

4. RELEVANT LICENCES USED IN THE DEMONSTRATOR

The Middleware has been implemented using Open Source license agreement. It is formed by different Open Source software packages with their corresponding license agreements. A list of the most relevant software packages and licences, used in the development of the Middleware, are summarized below:

- Flask and PythonEve: Berkeley Software Distribution (BSD).
- MongoDB: Server Side Public License (SSPL)
- Apache Hadoop: Apache Licence V2 (Apache-2.0)
- R: R as a package is licensed under GPL-2 | GPL-3. File doc/COPYING is the same as GPL-2. Some files are licensed under 'GPL (version 2 or later)', which includes GPL-3. Some header files are distributed under LGPL-2.1. The following licenses are in use for R or associated software such as packages.
    - The "GNU Affero General Public License" version 3
    - The "Artistic License" version 2.0
    - The "BSD 2-clause License"
    - The "BSD 3-clause License"
    - The "GNU General Public License" version 2
    - The "GNU General Public License" version 3
    - The "GNU Library General Public License" version 2
    - The "GNU Lesser General Public License" version 2.1
    - The "GNU Lesser General Public License" version 3
    - The "MIT License"
    - The "Creative Commons Attribution-ShareAlike International License" version 4.0

In addition to this background licences, the Middleware will be released as Open Source software framework under the European Union Public Licence v.1.2 (EUPLV1.2). There are no incompatibilities detected between the EUPL and the licences of the background software packages.

5. OVERVIEW OF THE MIDDLEWARE

The Middleware demonstrator has been built in the servers managed by CIMNE. These servers provide all the functionalities required to assure a seamless communication with the different FLEXCoop components.

Figure 2 shows a more detailed overview of the Middleware framework. As it can be seen, the Middleware is formed by the following main components (from left to right):

- **Open ADR interface:**

This communication layer enables the communication among the Middleware and the OSBs installed at the households. This communication layer follows the Open ADR communication protocol.

- **Big Data analytics engine:**

This component is the engine of the data storage and data analytics modules. It is formed by a distributed database (HBase), a high performance NoSQL database (Mongo DB), a data analytics framework (Hadoop) and a task management and scheduler (Celery).

- **Extract, Transform and Load (ETL) system.**

ETL is the general procedure of copying data from the external source into the distributed database. This procedure is done in the three different steps: (i) extract data from the heterogeneous sources; (ii) transform data by applying data cleansing or transforming it into a different format/structure for the purposes of querying and analysis. Finally, (iii) load data into the distributed database, ready to be analysed.

- **OAuth authorization framework:**

The OAuth 2.0 authorization framework enables a FLEXCoop component to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the FLEXCoop component to obtain access on its own behalf

- **API-REST interface:**

It is the communication interface between the Big Data analytics engine and the Client REST of the FLEXCoop components. The API is fully developed following the REST standard.

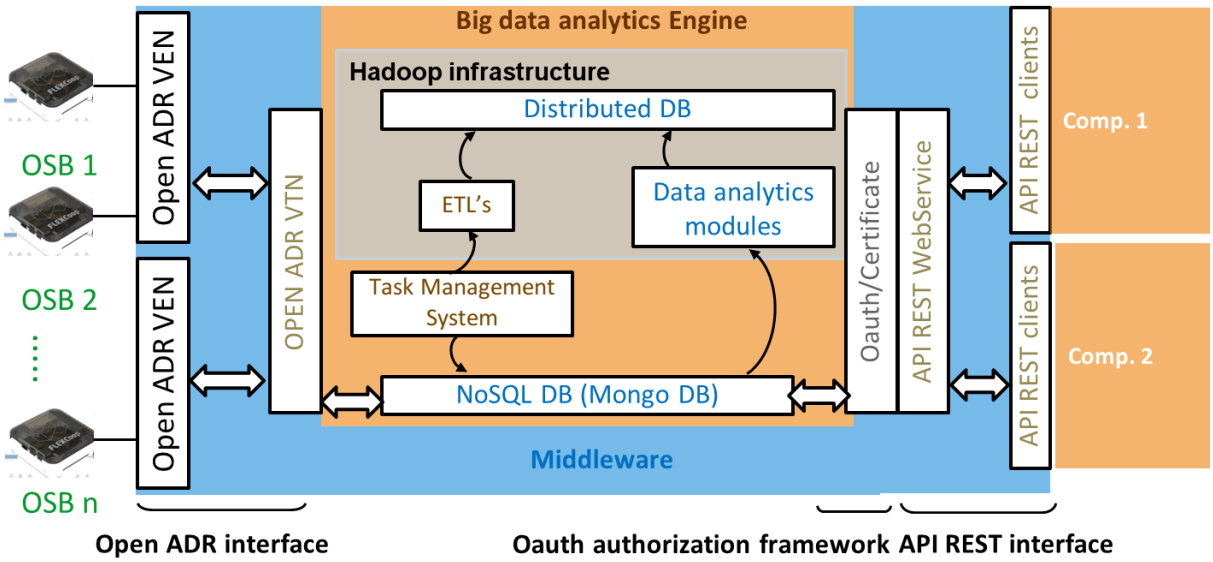

Figure 2: Middleware and Big Data Architecture

5.1. OpenADR interface.

The OpenADR 2.0b protocol is implemented with the aim to provide the Middleware with an open standard of communication with any kind of household electronic device or communication gateway. Within the FLEXCoop project, this implementation has been tested with the OSBs which are installed in the pilot sites.

OpenADR 2.0b is an open and interoperable information exchange model which standardizes the message format used for Auto-DR. The concept of an open specification is intended to allow anyone to implement the two-way signalling systems, providing the servers that publish information to the automated clients subscribing to the information. This standard covers the signalling data models and includes information related to specific Demand Response (DR) electric reduction or shifting strategies, which are taken at the facility. This standard can be leveraged to manage customer energy resources, including load, generation and storage, via signals provided by grid and/or market operators. These resources may be identified and managed as individual resources with specific capabilities, or as virtual resources with an aggregated set of capabilities.

OpenADR 2.0b uses a Service-Oriented Architecture (SOA) in which all interactions occur between entities called: (i) Virtual Top Nodes (VTNs) and (ii) Virtual End Nodes (VENs). The VTN acts as the master in the communication protocol, It is implemented in the Middleware and is the communication actor that sends the DR events to the VEN's which are connected to it. The VEN is the slave part of the communication protocol; it will send the metering reports with collected data and will receive the information of the DR events it has been selected to participate. To communicate using OpenADR 2.0b, each component shall implement the VEN, providing a HTTP or XMPP server to send and receive the data.

The demand-response signals make the VTN interact with a VEN in order to influence or change the load profiles of the demand-side loads, associated with the VEN in question. Two types of signals can be used: prices and load dispatches. The prices might be used if the objective is to incentivize the demand-side resource to change the load profile with a price incentive, thus implicitly. In a load dispatch signal there is an explicit instruction on what the load profile should be.

Two different protocols can be used when communicating with OpenADR: HTTP and XMPP. In the Middleware implementation, only the HTTP solution is implemented, while the XMPP will be ready for the final version.

The communication will be secured by using a client RSA certificate that will be obtained by the OSB on installation, using the final user's OAuth2.0 authentication. (See FLEXCoop D.4.5 on FLEXCoop Security Access Control (SEAC) Framework – Preliminary Version).

The OpenADR 2.0b profile supports the following services:

- **EiEvent:** To notify the VENs of upcoming DR events and sending DR signals from VTN to VEN
- **EiOpt:** Opt-in and opt-out capability by the VEN
- **EiReport:** Specifies the report by the VEN to the VTN; typically supports the VTN's prediction and monitoring capabilities
- **EiRegisterParty:** Establishment of communication between a VEN and a VTN.

5.1.1. EiEvent service

This service manages automatic demand response (ADR) events. The Open ADR 2.0b protocol defines two different methods (PUSH and PULL):

1. PUSH method**:** In Figure 3 a data flow diagram is shown. The communication procedure is as follows:
    a. The VTN starts the communication.
    b. The VTN sends the events to the VEN using **oadrDistributeEvent**, which can contain one or many **oadrEvent**.
    c. The VEN responds with **oadrCreatedEvent** if required by the request. This request must contain an **eventResponse** matching each Event

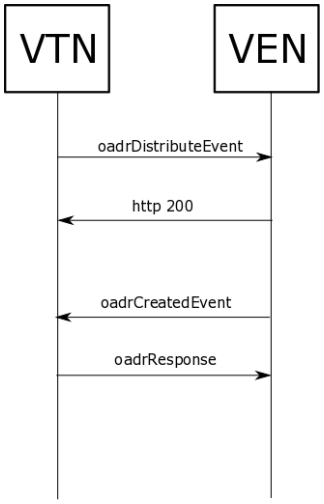

Figure 3: Push method for the EiEvent service

2. PULL method: A data flow diagram is shown in Figure 4. The procedure is as follows:
    a. The VEN starts the communication which can include a periodic request

b. The VEN sends **oadrPoll** to the VTN
c. The VTN responds with **oadrDistributeEvent**
d. It Continues like the PUSH method
e. The VEN can also send one-time **oadrRequestEven**t instead of **oadrPoll**

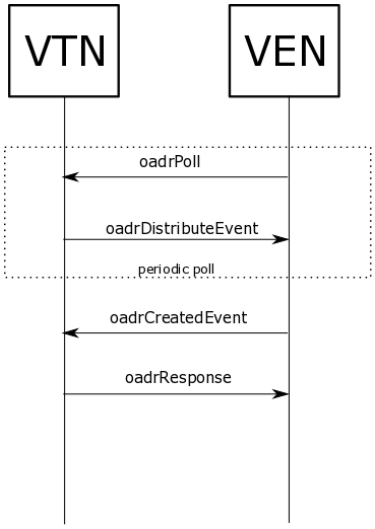

Figure 4: **Pull method for the EiEvent service**

5.1.2. EiReport service

This Service is designed to send and receive reports. In OpenADR 2.0b there are different types of Reports:

- **METADATA Reports:** used to specify reporting capabilities
- **DATA Reports:** actual data that may be measured or calculated

In the report service, the VEN and VTN can register to each other's reports in order to exchange information. This are the available operations:

- **Registering Reporting Capabilities:** One node (VEN or VTN) sends the reporting capabilities to the other node. (using METADATA REPORT)
- **Requesting Reports:** One node, requests to another node for a report
- **Sending Reports:** The node sends the requested report
- **Cancelling Reports**: The node cancels the generation of ongoing and periodic reports.

5.1.3. EiRegisterParty service

This service is designed to support the registration of VENs with VTNs. The VEN needs to register to the VTN in order to start the communication. The registration procedure follows the next steps:

1. The VEN sends the **oadrCreatePartyRegistration** by sending its information to the VTN.

2. The VTN responds with **oadrCreatedPartyRegistration** containing the profiles and

information about the registration.

3. From now on. The VEN will have a VEN-ID and will be registered to the VTN.

5.1.4. EiOpt service

This service was designed to create and communicate Opt-In and Opt-Out schedules from the VEN to the VTN. **EiOpt** is created using **oadrCreateOPT** and is used to communicate the availability of the node. Its main objectives are:

- To communicate to the VTN a period of temporary availability for a specific set of **eiTargets**
- To communicate to the VTN a period of temporary unavailability for a specific set of **eiTargets**
- To **optIn** or **optOut** of a previously acknowledged event for a specific set of **eiTargets**

The VEN may at any time cancel a temporary availability schedule by using the **oadrCancelOpt.**

More detailed information about OpenADR protocol and XML Schemas can be found in https://www.openadr.org/specification-download.

5.2. RESTful API interface

The API is fully developed following the REST standard. This component is the FLEXCoop components gateway to communicate internally between all the components and with the Big Data analytics engine. The main objectives of the API are: (i) interconnect all the FLEXCoop components to allow internal communication; (ii) enable the centralized storage platform defined in the architecture; and (iii) allow the input/output of data into the analytic modules. The **RESTful** API communication interface has been developed using the **PythonEve** framework that provides an easily customizable REST API, which can be modified by the developers by defining the resources in a simple JSON document. Figure 5 shows an example of a defined resource to manage a time series of temperature.

```
{
  "item_title": "temperature",
  "resource_methods": ["GET", "POST"],
  "aggregation":{
    "index_field":"ts"
  },
  "schema": {
    "ts": {
      "type": "datetime"
    },
    "value": {
      "type": "float"
    }
  }
}
```

Figure 5: JSON document for a resource

In the previous definition, we can see the "schema" field indicating that the resource will be composed of a "ts" field containing a datetime and a "value" field containing a float.

Moreover, the definition allows to **GET** (retrieve the documents), **POST** (uploads new documents). **PATCH** (update an existing document) and **DELETE** (remove an existing document) are other possible actions that are not allowed in the time series resource.

Every component will be required to provide the definition of these simple resources configuration.

Once the resource has been defined, all FLEXCoop components will be able to request this resource if the user credentials and permissions match the requested document. This way, the communication between the components will be possible.

The authentication of the Users is done using the access token provided by an authorized OAuth2.0 server ("See D4.5").

5.3. Big Data analytics engine

The Big Data analytics engine has been developed on the basis of the background Open Source Big Data platform **ENMA (Energy That Matters)**. ENMA is a Big Data platform with the following features:

- Mass storage and management of unstructured data
- High speed database (Mongo DB) and distributed database (Hbase)
- Easy-to-implement data analysis layer (R and Python)
- Layers of communication based on SOA architecture with API-Restful

ENMA has been commercially validated with interactive data driven energy awareness services offered to 350,000 users of electricity trader companies and within the EU-H2020 projects: EMPOWERING, SIM4BLOCKS, FLEXCOOP. A commercial version (BEE Data) is being marketed by BEE Data Analytics (https://beedataanalytics.com/) since May 2017. It is available under the Open Source EUPL V1.1 license.

All the new features developed within the FLEXCoop project are included in the second upgraded version of ENMA. The software code with the new capabilities can be found in the FLEXCoop repository through the links of section 7.

This architecture allows the storage and wrangling of large amounts of data. This is made up by a combination of low-cost hardware and database technologies that allows the acquisition, allocation and extraction of data to be processed in a distributed cluster. Essentially, the data collected, is stored in the different DB available in the Middleware depending on its nature. Data coming from the OSB's sensors, timeseries and data used to perform data analytics will be stored in the Hadoop Distributed File System (HDFS).The rest of the data will be stored in the MongoDB, as it is a fast read-write database available through the REST API.

The Big Data analytics engine is entirely developed using open-source software. From Figure Number 2 it can be seen that it is composed by the following software components:

1. Hadoop infrastructure.

2. Task Management System
3. NoSQL Database

5.3.1. Hadoop infrastructure

Apache Hadoop is an open-source framework that provides tools for distributed storage and processing. It allows organizations to process and analyse large volumes of unstructured and semi-structured data, heretofore inaccessible, in a cost and time-effective way.

Apache Ambari is used in order to manage the Hadoop cluster. It allows nodes to be added and removed to the cluster, install new components in existing working nodes, monitoring the cluster, etc.

The two main Hadoop components are **YARN** and **HDFS**:

1. HDFS (Hadoop Distributed File System): It consists of slave components called **DataNodes** where data is physically saved and a master process called NameNode that is responsible for maintaining the file system directory tree and has the information of where data effectively is (i.e. which blocks are available in every DataNode). All **HDFS** reads and writes are managed by **DataNode**.
2. YARN (Yet Another Resource Negotiator): It is responsible for processing Map-Reduce tasks using the master-slave paradigm. It consists of the ResourceManager (master similar to **NameNode**). It is in charge of managing the launched tasks. The NodeManager resides in the slave nodes. It receives Map or Reduce orders from the ResourceManager and executes those tasks in **YARN** containers.

There are many high level applications running on the main components. Two of them are used in this project:

- Hbase: Distributed key-value database. Provides real time read/write access and is built on top of **HDFS**. **Hbase** is used as the distributed big-data DB.
- Hive: Data warehouse on top of **HDFS**, which provides SQL-like querying which are translated into MapReduce functions. Hive queries are written in HiveQL, an SQL-like language.

Beside this, two different type of data analytic tasks can be differentiated depending on the nature of its process

ETLs

Extract, Transforms and Load (ETL) tasks are used for aggregate, clean and transfer the data from the Input/Output NoSQL database to the distributed HBASE database, pre-processing the data to ensure the good quality and format.

5.3.2. Data analytics modules

Once the information is stored in the distributed DB, asynchronous analytical modules are implemented to generate the needed results for the services offered to the utility. The technologies used for the algorithms are a combination of R, Hive, and Python software libraries using the Map Reduce paradigm to allow complex calculations over large sets of data.

R is an open-source programming language for statistical computing. In order to use R in the Hadoop environment **Rhadoop** packages is being used. This package offers access to the distributed DB and facilitates the implementation of Map Reduce algorithms using common R functionalities.

Python can also be used in the same manner with the **MRjob**. Python scientific libraries, such as **Pandas**, **SciPy** or **NumPy**, enable other advanced means for data analysis as an alternative to R. The combination of these languages allows the use of the most highly optimized implementations according to the requirements of the algorithm and this generates less development effort and a shorter data processing time when the code is executed.

Additionally, the **rpy2** package can allow to integrate Python map-reduce jobs and R analytical scripts in an easy way.

5.3.3. Task Management system

This component is in charge of scheduling and synchronizing the tasks in the Big Data analytics engine by means of **RabbitMQ** and **Celery**. In essence, the scheduler picks up the new task to be executed in according to a scheduling policy. The **First In First Out** (**FIFO)** policy was chosen because the batch operation of the tasks made other variants (like Round Robin), frequently applied in time-sharing environments, inefficient. **Celery** is the scheduler itself. **RabbitMQ** is a fast internal message queuing system used to interchange information between tasks with different paradigm technologies.

5.3.4. NoSQL Database

To allow the users and components access to the Big Data analytical modules, it is necessary to introduce an Input/Output Gateway. This Gateway is implemented with the NoSQL MongoDB. **MongoDB** is a cross-platform document-oriented database, classified as **NoSQL**, that uses **JSON-like** documents.

The main difference between an SQL database and a noSQL database is that while SQL are relational databases structured in tables, noSQL does not have a fixed structure, allowing to save high amounts of data and making the database easily scalable and replicable. This kind of databases have emerged as a backend to support Big Data applications.

The installation of MongoDB in an SSD drive makes it perfect for having fast access to the stored data for input/output operations.

6. Integration with other FLEXCoop components

As it is mentioned in previous sections, the communication between the Middleware and all the FLEXCoop components will be performed through the RESTful API interface. The DER management system and the component, which interprets the IEC61850 protocol, will also use the RESTful API. The communication between the Middleware and the OSBs will be performed through the Open ADR protocol.

However, there are two components, which will have a stronger integration with the Middleware:

- Generation forecasting modules: The mathematical basis of these modules are described in FLEXCoop Deliverable D.3.1 on Models of DER Devices and associated Forecasting

Algorithms. These modules will be incorporated as new data analytics modules within the data analytics framework of the Big Data analytics engine.

- DER Registry: This component will be installed at the same servers than the Middleware and the Big Data analytics framework. In this preliminary version, the communication with the Middleware will be made through the RESTful API interface, however, in the second version, more integrated communication procedures will be analysed.

The integration of the Middleware and the FLEXCoop components will be performed in the Task 6.4 on Integration of FLEXCoop Components, Preliminary Testing, Parameterization and Pre-Pilot Validation.

7. PROGRAMMING LANGUAGE

All the development has been performed using Python 3, using different frameworks to make the development of the functionalities more easy and secure. The **RESTful** API is implemented using the framework **Python Eve**. The OpenADR communication has been programed using **Flask** for the HTTP server. For the **XMPP** server that is still under development the **xmppFlask** framework will be used.

8. SOURCE CODE OF THE RELEASE

All the resource code can be found in the GitLab provided by Fraunhofer FOKUS to share the code between the partners. The Middleware is under the EUPL licence as stated above.
The links to the software repositories where the source codes are upload, are listed below:

- Middleware REST:
  https://gitlab.fokus.fraunhofer.de/FlexCoop/Middleware_rest
- OpenADR:
  https://gitlab.fokus.fraunhofer.de/FlexCoop/openADR_Middleware
- ENMA Architecture:
  https://github.com/BeeGroup-cimne/ENMA

9. INSTALLATION GUIDE

Both Middleware REST and Middleware OpenADR have very simple installation guide available in the README.txt file of the FLEXCoop software repository:

9.1. RESTful API interface

1. Create a virtualenv using python 3 and install the required packages

   ```
   virtualenv -p python3 venv
   source venv/bin/activate
   pip install -r requirements.txt
   ```

2. Create a file with the environment variables

   ```
   export MONGO_HOST='<mongo_host>'
   ```

```
export MONGO_PORT=<mongo_port>
export MONGO_USERNAME='<mongo_username>'
export MONGO_PASSWORD='<mongo_password>'
export MONGO_DBNAME='<mongo_database>'
```

3. Run the server

```
python run.py
```

9.2. OpenADR interface

1. Create a virtualenv using python 3 and install the required packages

```
virtualenv -p python3 venv

source venv/bin/activate

pip install -r requirements.txt
```

2. Create a virtualenv using python 3 and install the required packages
3. Create a file with the environment variables

```
export HOST=''
export PORT=''
export VTN_PREFIX=''
export VEN_PREFIX=''
export MONGO_URI='mdb://<usr>:<pass>@<host>:< port>/<db>
```

Run the server

```
python app.py
```

10. REQUIREMENTS COVERAGE

Open ADR interface	Implemented. Communication tests with the OSB in process.
Big data analytics Engine - Hadoop Infrastructure - Task Management system - NoSQL database	 Implemented Implemented Implemented
RESTful API interface	Implemented. Communication tests with the FLEXCoop components in process.
OAuth2 / OpenID	Implemented the authentication client. Communication tests with the OSB and the FLEXCoop components in process.

Table 1: Requirements coverage

11. DEVELOPMENT AND INTEGRATION STATUS

Current Status	Preliminary demonstrator
Development status	Second version under development
Pending development actions	Implementation of the XMPP protocol and of some OpenADR messages
Integration status	Open Marketplace already integrated PV generation forecasting models integration in process
Pending integration actions	Pending integration with all the FLEXCoop components

Table 2: Development and integration status

12. CONCLUSION

With this demonstrator we provide an early view on the Middleware and its integration into the FLEXCoop solution. The main objectives of the Middleware are available and a preliminary version has been implemented.

The software source codes of the preliminary version of the Middleware have been made available to the FLEXCoop partners through the software repository managed by Fraunhofer FOKUS.

Besides, the OpenADR protocol and the RESTFul communication layers are already implemented and made accessible to the FLEXCoop developers so that they can implement the client side element within their FLEXCoop component.

The security and access control platform has been already integrated into the Middleware and made available to the FLEXCoop developers. The OAuth2 protocol has been used to implement this platform linked to the D4.5.