Project acronym: **DATAZERO**

Project full title:

DATAcenter with Zero Emission and Robust management using renewable energies

D5.2: Middleware of DataZero

Author: J.-M. Nicod

Version: 1.0

Date: 12/03/2018

Deliverable information

Deliverable number	D5.2
Contractual date of delivery	31/03/2018 (M30)
Actual date of delivery	12/03/2018
Title of deliverable	D5.2: Middleware of DataZero
Dissemination level	Restricted to other program participants (including the ANR Services)
WP contributing to the deliverable	WP5
Author	J.-M. Nicod
Co-authors	A. Sayah, P. Stolf, S. Caux, J.-M. Pierson, G. DaCosta, L. Philippe, G. Baudic, G. Rostirolla

Revisions

Version	Date	Author	Comments
1.0	31/03/2018	J.-M. Nicod	Final version

Abstract

The aims of this deliverable are to:

- Describe the middleware that makes the link between DataZero modules and makes possible the management the IT part and the Electrical part of a datacenter;
- Describe the extra-modules that help the user to supervise that datacenter.

The deliverable of this WP is a the software and the role of this document is to support the useful descriptions to understand our implementation choices.

Keywords

TODO

1. Global system structure

The global structure of the Datazero system is written within deliverable D3.1. entitled “Interactions between system modules and messages format”.

The next figure gives an overview of the middleware architecture on which it is possible to see the different components that are connected to the middleware and the messages that can be exchanged between components.

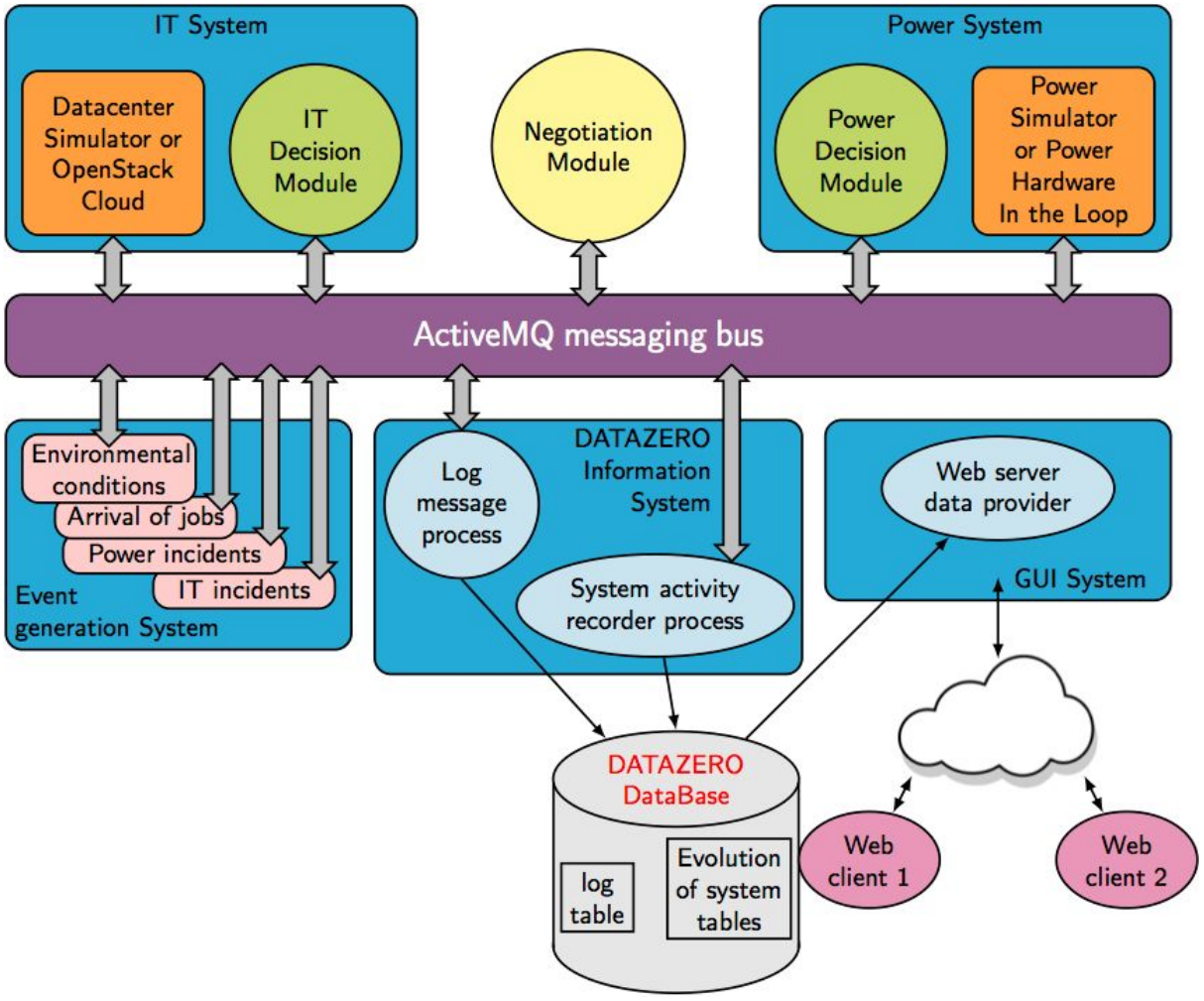

Fig 1. Middleware overview

The main objective of this deliverable is to ease the deployment of the Datazero architecture and to help the implementation of different modules that are connected with the middleware. This report will also help future developments for the Datazero next generation.

The deliverable is presenting packages and associated processes to allow the implementation of the Datazero software part. This report is organised as follows:

- Short description of software components that are developed and the list of functions which have to be implemented within each software component.
- These processes are described considering the current implementation. Some modules are under development because all the tasks of the project are not completed yet. However, this report allows these modules be connected to the system without challenging the middleware.
- Description of what one needs to install the Datazero middleware.

2. Software components

The Datazero middleware consists of several packages. The name of the source files that are included within that packages are very explicit. Most of them have already been described within Deliverable 3.1. and its appendix that describes exchanging messages ActiveMQ Messaging Bus. So the only files that have not been described yet are described here.

2.1. Package common

This package contains useful classes and the root of Datazero class hierarchy.

DataZeroObject.java	This class is the root of the DataZero class hierarchy. Every class has DataZero as a superclass
DataZeroObjectSet.java	Set of DataZero objects
TimeStampedArray.java	List of values associated with a timestamp
Profile.java	Power profile exchanged during the negotiation phases
Assoc.java	Generic key-value pair to help prepare data for the GUI

2.2. Packages description

These packages describe the constitution of the managed datacenter.

2.2.1. Package description.power

This package describes the different managed power sources.

SourceType.java	Different types of source
SourceState.java	Different states of source

AbstractSourceDescription.java	Common characteristics of any type of source
BatteryDescription.java	Specific description of batteries
PhotoVoltaicDescription.java	Specific description of solar sources
SuperCapacitorDescription.java	Specific description of supercapacitor sources
WindDescription.java	Specific description of wind turbine sources
GridDescription.java	Specific description of electricity supplier
FuelCellDescription.java	Specific description of fuel cell sources
DataCenterPowerDescription.java	full description of the datacenter electrical equipment

2.2.2. Package description.it

This package describes managed IT resources.

MachineState.java	Different state of machine
RackState.java	Different state of rack
RackDescription.java	Description of one rack
MachineDescription.java	Description of one machine
Flavor.java	Description of cpu, ram and disk resources needed
DataCenterITDescription.java	Description of IT part of Datacenter

2.2.3. Package description.datacenter

This package describes totally the datacenter (electrical and IT points of view).

DataCenterDescription.java	Electrical and IT description of Datacenter

2.3. Packages activity

These packages describe datacenter activity: orders given to its various components, their states, encountered events, ...

2.3.1. Packages activity.it

These packages describe datacenter activity at different IT levels: machines, racks and globally at DC level.

2.3.1.1. Package activity.it.resources

MachineActivity.java	Information about machine activity: state, frequency, power used and statistics on the states of its jobs
RackActivity.java	Information about rack activity: state, power used and statistics on the states of its machines and on the states of the jobs assigned to the machines of the rack
DataCenterITActivity.java	Information about DC activity: power used and statistics on the states of its racks, machines and jobs

2.3.1.3. Package activity.it.jobs

This package provides classes for managing jobs submitted to Datazero (arrival, placement, terminaison ...).

JobState.java	Different states of a job
JobTypeResource.java	Different types of resources that a job can use (memory, cpu, disk, ...)
JobResources.java	Resources required at a given moment
JobPhase.java	Phase concept used to describe resource requirements of a job over time
JobDataZero.java	Class to represent a job in Datazero: arrival time, resources needed, image to run, ...

2.3.2. Package activity.power

These packages describe datacenter activity at electrical level: electrical environment and power sources.

2.3.2.1. Package activity.power.environment

This package describes the environment associated with green sources (wind or solar) and production forecasts.

SourceEnvironment.java	Wind and solar environment, with forecasts (wind speed and sunshine and estimated production) and observation of real values

2.3.2.2. Package activity.power.sources

This package provides classes that allow to observe the activity of the electric sources and to control their modes of operation.

SourceDiagnostic.java	Getting source characteristics
PowerSourceActivityData.java	Managed informations about the electrical production of a source: state, observed, lost and cost production
SourceProduction.java	Information concerning the electrical production of a source for a given period of time

2.3.2.3. Package activity.power.datacenter

This package provides classes that allow to observe the activity of the power Datacenter system.

DcPowerObservation.java	Managed information about the electrical consumption of the Datacenter: observed, lost and cost production of renewable sources and consumption and cost of grid use
DataCenterPowerActivity.java	Information concerning the electrical consumption of the Datacenter for a given period of time

2.3.3. Package activity.datacenter

This packages describes the overall activity of the datacenter (IT and electrical).

DataCenterActivity.java	IT and electrical activity of the Datacenter

2.4. Package log

This package provides real-time access to data logged by Datazero system. The data can be extracted according to different criteria (topic, timestamp, more recent, etc.).

LogRequest.java	Description of the desired extraction
LogResponse.java	An element of the answer
LogResponseSet.java	Set of all the messages respecting the given criteria

2.4. Package database

The datazero database, implemented using MariaDB, includes several tables specifically fed with data extracted from certain messages describing the system or its behavior. Each table has two features: adding new records and selecting records that match given criteria.

Common and useful classes	
DataZeroDB.java	DataZero database representation
TableDB.java	base class inherited by all classes representing datazero tables
ListOfLists.java	Structured response to a select , in the generic form of a list of lists.

Log table	
TableLog.java	Table that stores all messages exchanged on the ActiveMQ messaging. It is used by the activity.processes.log process described below.

Common electrical sources or IT resources classes	
TableState.java	Generic class that memorize state changes, instantiated for machine states, racks states and electrical sources states.

TableChange.java	Class that allows to create two tables, one that memorizes the frequency changes of the machines or the other one for the power changes of the electric sources.

Power tables	
TableSourceEnv.java	forecasts and measures really observed for wind speed and sunshine.
SourceTableActivity.java	Electrical sources activity (production, lost, cost, state).
DcPowerTableActivity.java	Summary of the power activity, by source type

IT tables	
MachineTableActivity.java	Observation of the activity of the computers over the time (state of the machine, states of its jobs, ...)
RackTableActivity.java	Observation of the activity of the racks over the time (its state, state of its machines, states of its jobs, ...)
DcItTableActivity.java	Observation of the IT activity of the datacenter over the time (state of its racks, state of its machines, states of its jobs, ...)

Negotiation tables	
TableNegotiation.java	Table that records all negotiation messages exchanged between ITDM, PDM and Negotiation processes. Each negotiation has a unique identifier.
ResponseGetNegotiationList.java	Class that retrieves the list of registered negotiations, especially for the GUI
ResponseGetNegotiationSelect.java	Class that retrieves messages related to given negotiations

2.5. Package negotiation

This package provides a set of classes that can be used to perform the negotiation process between the PDM and the ITDM.

Negotiation.java	Each negotiation will be represented in the system by an object of this class.
ProfileSet.java	A profileset object describes a power profile proposed / requested by one of the protagonists of a negotiation
ProfileRequest.java	This class allows the negotiation process to request the ITDM or PDM to send a power profile
ProfileResponseITDM.java	During a negotiation, this class describes the profile sent by the negotiation process to the ITDM process to guide its future proposals
ProfileResponsePDM.java	During a negotiation, this class describes the profile sent by the negotiation process to the PDM process to guide its future proposals
NegotiationResponseType.java	This class describes the different responses provided by the negotiation process during a negotiation step (completed, new step, etc.).
NegotiationResponse.java	This generic class represents the response developed by the Negotiation process to the ITDM or PDM process (type+profile)

2.6. Package message

The message package has been described in deliverable 3.1 and the associated appendix.
We complete in this deliverable the list of messages by specifying their transmitters and receivers (see appendix).

2.7. Packages activity.processes

These packages implement the different systems described in the figure summarizing the middleware.
These systems are materialized by different multithreaded processes exchanging messages in a distributed system, via ActiveMQ messaging bus.
Some of these processes are in their almost definitive realizations, others are destined to evolve.

2.7.1. Package activity.processes.common

This package defines base classes that implement different exchange schemes via activeMQ and that can be inherited to easily define Datazero processes without worrying about ActiveMQ specifics.

DZProcess.java	Base class for a DataZero process hierarchy using activeMQ to communicate
DZMessageProducer.java	Datazero thread generating messages of a given topic (sender)
DZMessageConsumer.java	Datazero thread consuming messages of given topics (receiver)
DZMessageProducerWithReply.java	Producer thread that expects a response to the sent message (send/receive)
DZMessageConsumerWithReply.java	Consumer thread that sends a response to the received message (receive / send)

2.7.2. Package activity.processes.itdm, activity.processes.pdm and activity.processes.negotiation

These processes are the subject of specific studies whose results will be incorporated into the skeleton of processes programmed to illustrate the overall operation of the Datazero system.

2.7.5. Package activity.processes.openstack

TODO Eaton ?

2.7.6. Package activity.processes.power_system

TODO ?

2.7.7. Package activity.processes.event_generator

This process simulates the arrival of events that can impact the operation of the data center, both from an electrical and IT standpoint.

To be able to enrich it progressively and to cover little by little the whole of the external interactions, this process includes several threads, each one in charge of the triggering of particular events.

To allow these threads to transmit the events they generate to the datazero processes that must take them into account, some specific topics have been added. Currently, we have listed 4 exciters of the datazero system:

- ITGenerator threads that generate messages about the state of IT resources (shutdown of machines or racks, restart, etc.). The topics identifying these messages are EVT_MACHINE_CHANGE_STATE and EVT_RACK_CHANGE_STATE.
- JobGenerator that simulates the arrival of new jobs by injecting messages into the system for the topic IT_JOB_ARRIVAL.
- EnvironmentInfosProducer that generates messages about the weather forecast (wind and sunshine) associated with the topic EVT_PREV_ENV
- SourceStatesProducer that produces messages about the state of the electrical sources (failure, restart, etc.), identified by the topic EVT_SOURCE_CHANGE_STATE.

These exchanges will be traced in the logs of the datazero system.

2.7.8. Package activity.processes.log

This process intercepts all messages transmitted on the ActiveMQ bus and stores them in the logTableDataZero table with their timestamps.
It is therefore concerned by all DataZero topics.

2.7.9. Package activity.processes.database

This process intercepts certain messages on the ActiveMQ messaging bus and specifically extracts data that are stored in specific tables, so that they can be retrieved on demand, especially at the request of the GUI.

Received topics and concerned tables

DataZero Topics	Concerned database table
IT_MACHINE_CHANGE_STATE	machineStateTable
IT_MACHINE_CHANGE_FREQUENCY	changeFrequencyTable
IT_RACK_CHANGE_STATE	rackStateTable
ELEC_SOURCE_CHANGE_STATE	sourceStateTable
ELEC_SOURCE_CHANGE_POWER	changePowerTable
IT_MACHINE_STATE	machineActivityTable
IT_RACK_STATE	rackActivityTable
IT_DC_STATE	dcItActivityTable
ELEC_ENV_PREVISION	wind sunshine
ELEC_SOURCE_PRODUCTION	sourcesTableActivity

ELEC_DC_STATE	dcPowerActivityTable
PDM_TO_NEGO	negotiationTable
ITDM_TO_NEGO	negotiationTable
NEGO_TO_ITDM	negotiationTable
NEGO_TO_PDM	negotiationTable
NEGO_REQUEST_TO_ITDM	negotiationTable
NEGO_REQUEST_TO_PDM	negotiationTable

2.7.10. Package activity.processes.webserver

This process is waiting for HTTP GET requests from the GUI. It generates the SQL queries to retrieve the requested data from the datazero tables and puts them in the format expected by the GUI.
All GET requests currently known are taken into account.

3. Middleware installation

Before deploying the middleware, one needs to ensure that the following prerequisites are installed on the system:

- Node.js >= 6.x
- MariaDB
- ActiveMQ
- Ant
- The geos library (package libgeos-dev on Ubuntu, geos on OpenSUSE and Fedora)
- JDK 1.8
- Python 3.5 or above with pip3 (for the negotiation)
- Python 2.7 and Gurobi for the PDM

The middleware is known to work on Linux and Mac; Windows may work too but is not tested.

Additionally, the following jar files are required: ActiveMQ and MariaDB connectors, Jackson and Google GSON. They can be found in DZMaquette/usr/share/java, except the ActiveMQ Java connector jar which lies in DZMaquette/usr/share/apache-activemq-5.14.4.
The OpenStack interface requires the dependencies available in DZMaquette/usr/jclouds.

Configuring and building

In the DZMaquette directory, type

```
ant build
```

In the negotiation_prototype directory, type

```
pip3 install -r requirements.txt
python3 setup.py install --user
```

GUI

To use the GUI, you first need to specify in the globals.ts (found in DZMaquette/src/gui/DZServer/src/app) file the IP address or hostname of the machine running the server backend. The default value is localhost. Then, use `npm install` to get all dependencies, and `npm start` each time you need to run a development server. The GUI will then be available at http://localhost:4200/.

Note: depending on your version of Node.js (only 6.x seems to be affected), there may be some errors while installing packages about "unresolved peers". If this happens, it is likely that npm is trying to install package versions which are too recent: the workaround is to force installation of the right version of the packages in error one by one, using `npm install package@version`. The correct version number to use can be found in the package.json file, ignoring the ^ character.

Database

Initialize the database used by the log system and the GUI, by creating the appropriate database, user and tables. The commands can be found in the deployment/init_db.sql script. This step only needs to be done once. Some of the lines at the beginning of the file assume that the MariaDB root user does not have a password set, and can be commented out if this is not the case.

Running the middleware

- Make sure that the MariaDB daemon is running. If not, run
  `systemctl start mariadb.service`
- Start ActiveMQ. STOMP protocol support has to be enabled, this page has more information on how to enable it : http://activemq.apache.org/stomp.html.
  `activemq start`
- Start the negotiation example and dummy PDM, again in a new terminal:
  `cd {installdir}/negotiation_prototype/examples/demo_activemq`
  `python3 demo_02_2018.py --fake-pdm -a {activemqIP}`
  where `installdir` is the directory where you extracted the archive, and `activemqIP` is the IP address of the computer where ActiveMQ was started.
  Alternatively, you may want to write your own experiment file for the negotiation: see the provided examples and documentation for more information. In all cases, make sure that any real DM you want to use is started *before* the negotiation module.
- Start the LogDaemon, DataBaseDaemon and WebServer Java classes, each in a separate terminal.
  `ant RunLogDaemon`
  `ant RunDataBaseDaemon`

```
ant RunWebServer
```

- A generator for wind and sunshine prevision data is also available in DZMaquette/src/activity/processes/event_generator/EventGenerator.java. It can be used if necessary to fill the database with real weather data obtained on the first three months of 2018.

Messages, from sender to receivers

Topic	MessageContents	Sender	Receiver
IT_MACHINE_STATE	MachineActivity	Openstack DcWorms	ITDM DataBaseDaemon
IT_MACHINE_CHANGE_FREQUENCY	MachineChangeFrequency	ITDM	Openstack DcWorms DataBaseDaemon
IT_MACHINE_CHANGE_STATE	MachineChangeState	ITDM	Openstack DcWorms DataBaseDaemon
		Openstack DcWorms	ITDM DataBaseDaemon
IT_RACK_STATE	RackActivity	Openstack DcWorms	ITDM DataBaseDaemon
IT_RACK_CHANGE_STATE	RackChangeState	ITDM	Openstack DcWorms DataBaseDaemon
		Openstack DcWorms	ITDM DataBaseDaemon

Topic	MessageContents	Sender	Receiver
IT_JOB_PLACEMENT	JobPlacement	ITDM	Openstack DcWorms
IT_JOB_CHANGE_STATE	JobChangeState	ITDM	Openstack DcWorms
IT_JOB_TERMINATED	JobConsumptionOrTermina ison	Openstack DcWorms	ITDM
IT_JOB_CONSUMPTION	JobConsumptionOrTermina ison	Openstack DcWorms	ITDM
IT_INFOS_AFTER_JOB_EXIT	JobInfosAtTerminaison	Openstack DcWorms	ITDM

Topic	MessageContents	Sender	Receiver
ELEC_ENV_PREVISION	SourceEnvironment	PSProcess/PSEnvironmentObserver	PDM DataBaseDaemon
ELEC_SOURCE_PRODUCTION	SourceProduction	PSProcess/PowerProductionThread	PDM DataBaseDaemon
ELEC_SOURCE_CHANGE_STATE	SourceChangeState	PSProcess/SourcesManagerThread	PDM DataBaseDaemon
ELEC_SOURCE_CHANGE_POWER	SourceChangePower	PSProcess/SourcesManagerThread	PDM DataBaseDaemon

Topic	MessageContents	Sender	Receiver
PDM_TO_NEGO	ProfileSet	PDM	NEGO DataBaseDaemon
ITDM_TO_NEGO	ProfileSet	ITDM	NEGO DataBaseDaemon
			DataBaseDaemon
NEGO_TO_ITDM	NegotiationResponseToITDM	NEGO	ITDM DataBaseDaemon
NEGO_TO_PDM	NegotiationResponseToPDM	NEGO	PDM DataBaseDaemon
NEGO_REQUEST_TO_ITDM	ProfileRequest	NEGO	ITDM DataBaseDaemon
NEGO_REQUEST_TO_PDM	ProfileRequest	NEGO	PDM DataBaseDaemon

Topic	MessageContents	Sender	Receiver
EVT_MACHINE_CHANGE_STATE	MachineChangeState	EventGenerator/ITGenerator	Openstack DcWorms
EVT_RACK_CHANGE_STATE	RackChangeState	EventGenerator/ITGenerator	Openstack DcWorms
IT_JOB_ARRIVAL	JobDataZeroArrival	EventGenerator/JobGenerator	Openstack DcWorms ITDM
EVT_PREV_ENV	SourceEnvironment	Event_generator/EnvironmentInfosProducer	PSProcess/EnvironmentObserver
EVT_SOURCE_CHANGE_STATE	SourceChangeState	Event_generator/SourceStatesProducer	PSProcess/SourceObserver