Data Centres Optimization for Energy-Efficient and EnvironmentalLy Friendly INternet

Funding scheme: **Specific Targeted Research Projects – STREP Co-funded by the European Commission within the Seventh Framework Programme**

Project no. 609140

Strategic objective: **FP7-SMARTCITIES-2013 (ICT-2013.6.2)**

Start date of project: **October 1th, 2013 (36 months duration)**

Deliverable 4.2

Smart Grid Controller (Implementation)

Due date: 30/09/2015

Submission date: 23/10/2015

Deliverable leader: UCL

Author list: Stuart Clayman (UCL) , Alina Nistor (SIE), Alex Galis (UCL), Septimiu Nechifor (SIE), Evangelos Angelou (GRNET)

Dissemination Level

☒	PU:	Public
☐	PP:	Restricted to other programme participants (including the Commission Services)
☐	RE:	Restricted to a group specified by the consortium (including the Commission Services)
☐	CO:	Confidential, only for members of the consortium (including the Commission Services)

List of Contributors

Participant	Contributor
UCL	Stuart Clayman, Alex Galis
SIEMENS	Alina Nistor, Septimiu Nechifor
GRNET	Evangelos Angelou
IRT	Domenico Gallico, Matteo Biancani

Amendment History

Version	Date	Partners	Description/Comments
0.1	12/08/2015	UCL	Initial draft
0.2	16/09/2015	SIEMENS	ADR Server description, ADR message structure
0.3	23/09/2015	GRNET	Initial integration implementation contributions.
0.4	15/10/2015	UCL	Intro, Conclusions, editing
0.5	19/10/2015	SIEMENS	Document review
0.6	20/10/2015	IRT	Final Review
FF	23/10/2015	UCL	Final

Table of Contents

Figures Summary

Tables Summary

Abbreviations

ADR	Automated Demand Response
ADRS	Automated Demand Response Server
AMQP	Advanced Message Queuing Protocol
AP	Action Point
DC	Data Centre
DoW	Description of Work
DR	Demand Response
EC	European Comission
EP	Energy Provider
IPR	Intellectual Properties Rights
OpenADR	Open Automated Demand Response
PM	Person Months
PMO	Project Management Office
QCT	Quality Check Team
RES	Renewable Energy Source
SG	Smart Grid
SGC	Smart Grid Controller
TMC	Technical Management Committee
ToC	Table of Content
VEN	Virtual End Node
VTN	Virtual Top Node
WP	Work package
XMPP	XML Messaging and Presence Protocol

Executive Summary

This report is a companion document to the deliverable D4.2, which is the Smart Grid component implementation and presents the implementation attributes of the Smart Grid Controller. The Smart Grid Controller acts as the component that integrates the DOLFIN platform with the Smart Grid network. The purpose of this component is to be able to interact with the energy related requests from the Smart Grid side.

As such the design and implementation of an Automated Demand Response (ADR) Server are shown, together with the events from ADR server. These events are processed by the Smart Grid Controller and turned into a calendar of energy prices. The energy prices are passed onto other components of DOLFIN, which can use this information to optimize the consumption of energy.

The document presents the main interactions of this part of the DOLFIN system together with an overview of the implemented components and the messages they send. DOLFIN has built a Smart Grid Controller which exchanges messages with the Smart Grid network, is able to parse them, so as to create an internal representation of the message which is understandable by the DOLFIN platform, and acts as a gateway for interfacing with various Smart Grid networks.

1 Introduction

This report is a companion document to the deliverable D4.2, which is the Smart Grid component implementation and presents the implementation attributes of the Smart Grid Controller. The Smart Grid Controller acts as the component that integrates the DOLFIN platform with the Smart Grid network. The purpose of this component is to be able to interact with the energy related requests from the Smart Grid side.

The design of the DOLFIN architectural framework explicitly accommodates the energy management which is aimed at:

- Improving capital and operational efficiencies for DC operators through the use of a common organization, automation, and operations of all energy functions across the different domains, and
- Migrating from an ecosystem of separate energy management functions towards a coordinated arrangement of energy management functions.

As such, the Smart Grid Controller provides the mechanism for interacting with external energy providers and is a facilitator for the energy control loops within DOLFIN. In figure 1 we highlight, in RED, the parts of the DOLFIN architecture which are described within, namely the Smart Grid Energy Providers and the Smart Grid Controller.

The Smart Grid Controller is the component that integrates the DOLFIN platform with the Smart Grid network. The purpose of this component is to be able to interact with the energy related requests from the Smart Grid side.

DOLFIN is targeting the Smart Grid demand/response scenario, so as to maintain the electricity balance in the distribution network. The Smart Grid will be able to identify that an electricity stabilization issue is about to occur. In order to avoid such a situation, the Smart Grid network will issue a request, towards the DC, for lowering the power consumption. This could be expressed as an absolute figure, determining the exact level of power consumption that should be followed, or it could also be expressed as a percentage of decrease to the current levels of power consumption. The role of the Smart Grid Controller is to intercept and perform an initial examination of such message requests from the Smart Grid.

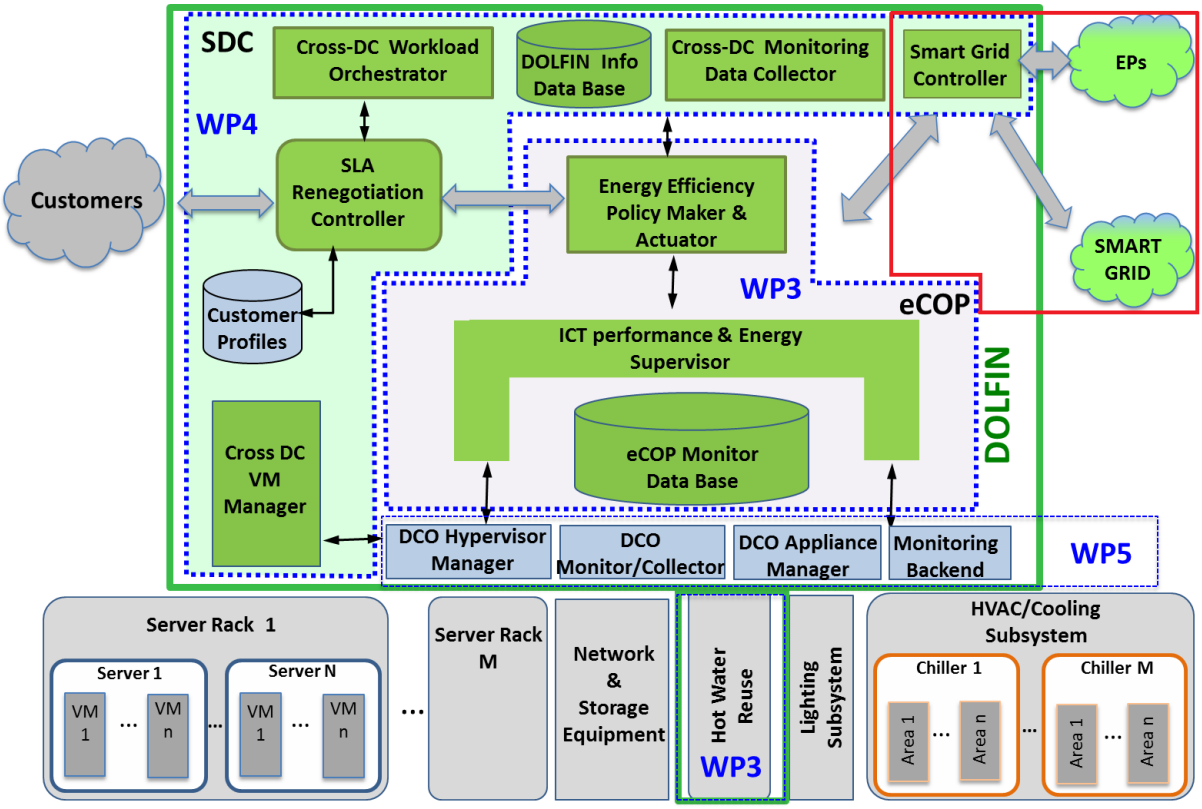

Figure 1: Overall DOLFIN architecture

In this respect, the Smart Grid Controller is required to exchange messages with the Smart Grid network and be able to parse them, so as to create an internal representation of the message that would be understandable by the DOLFIN platform. Furthermore, this component will have to incorporate the knowledge to communicate with possibly different protocols with the Smart Grid, thus acting as a kind of gateway for interfacing with Smart Grid networks.

The following sections of the document present the interactions, the design and implementation details of the highlighted components.

2 Smart Grid Controller

2.1 Module detail description

The Smart Grid Controller (SGC) acts as a gateway between the core DOLFIN infrastructure and sub-systems, represented by the Energy Efficiency Policy Maker & Actuator, and the outside world, namely the Smart Grid (SG) and the Energy Providers (EPs).

The SGC undertakes to support the DOLFIN System awareness of its energy surroundings, namely energy prices and SG status in terms of energy availability and (smart) energy grid stability. In this framework, it is responsible for assuring that the current and near future energy consumption profile of the DC are known to the SG and the EPs at all times. Last, it is responsible of informing the SG and the EPs of the availability of the DC's own energy resources, if any (e.g. in the case the DC is supported by one or more Renewable energy resources (RES)). The information from the DC side is provided by the Energy Efficiency Policy Maker & Actuator. The same holds for the information originating from the SG/EPs side; all is redirected to the Energy Efficiency Policy Maker & Actuator.

The data that are relayed to the Energy Efficiency Policy Maker and Actuator and the SG/EPs are lightly processed by the SGC in order to detect possible suboptimal situations that could lead to either less cost- and energy- efficient operation of the DC, or less stable SG operation, respectively. In that case, the interested party is responsible of issuing a proper action plan to optimize its place in the energy consumption/production chain. The SGC is not responsible for issuing such a plan; instead, it is responsible for communicating this information between the interested entities.

In short, the main tasks of the SGC are to:

1. Handle information related to the DC energy consumption and relay it to the SG/EPs;
2. Communicate to the SG/EPs information related to the DC's own energy sources, if available;
3. Handle information related to the energy tariffs from the SG/EPs side.

It should be noted that the SGC should be in place to support more than one communication protocols, in order to guarantee reliable communication with the set of cooperating SG Operators and the EPs.

For the communication between the nodes various protocols can be used, e.g. by using HTTP to receive or to send messages, or by using a message oriented middleware as XMPP (XML Messaging and Presence Protocol) or RabbitMQ [2]. In our specific case, we will be using the RabbitMQ protocol.

2.2 Smart Grid Controller Workflow

Every time a request is received by the SGC, it is translated in an internal SGC format (through the SGC- DOLFIN/SG/EPs interfaces) and is stored into the Requests/Responses Persistency Layer. Then, the request is retranslated and forwarded to the interested entity.

Upon reception of an answer, any information related to the DC/SG state, if any, is stored in the appropriate registers. Note that the DC state may comprise information related to its current and near future computing load, the availability of its own energy sources (if any), whereas the SG state may include data relate to the current energy tariffs or their trend, the availability of energy in the SG along with some grid stability metrics indicating possible instability in the near future.

Granted the previous DC/SG states, the SGC will evaluate whether an alert should be sent to any interested entity (DC/SG/EPs) along with the set of appropriate state information in order to render the receiver capable of performing sets of corrective actions to improve the DC cost- and energy-efficiency (from the DC side) or the SG stability (from the SG/EPs side).

In any case, the SGC acts as a mediation layer between the Energy Efficiency Policy Maker & Actuator and the SG/EPs side, also providing minor caching services (regarding the DC/SG states) in order to limit overhead, unnecessary traffic to/from the DOLFIN core internal ecosystem.

2.3 Energy Provider Interaction

Demand/response refers to mechanisms used to encourage consumers to reduce their demand and subsequently reduce also the peak demand for electricity. Electrical generation and transmission systems are sized to correspond to a predetermined peak demand, plus a margin for unforeseen events. By lowering the peak demand, we actually reduce the overall plant and capital cost requirements. Electricity Grid operators generally plan to use the least expensive generating plant (in terms of marginal cost) at any given period and use additional capacity from more expensive plants, as demand increases. Demand/response in most cases is targeted at reducing peak demand to reduce the risk of potential disturbances, avoid additional capital cost requirements for additional plant and avoid the use of more expensive and/or less efficient operating plant. On the other hand, consumers of electricity will also benefit by paying lower prices. To automate the process of demand response, technologies are employed to detect the need for load shedding, communicate the demand to participating consumers, automate load shedding and verify compliance with demand/response programs. To this respect and considering also that DCs are big consumers of electricity, it is reasonable that Smart Grid operators would like to apply demand/response mechanisms across DCs.

Our approach for interfacing with the Smart Grid is to utilize Open Automated Demand Response (OpenADR), which is a research and standards development effort for energy management. OpenADR is a communications data model designed to facilitate sending and receiving demand/response signals from a utility or independent system operator to electric customers. The intention of the data model is to interact with building and industrial control systems that are pre-programmed to take action based on a demand/response signal, enabling a demand/response event to be fully automated. The OpenADR specifications are designed to facilitate common information exchange between a utility and their end-use participants. During a demand/response event, the utility provides information regarding what has changed and on what schedule, such as start and stop times. The OpenADR specification defines the interface to the functions and features of an Automated Demand Response Server (ADR Server) that is used to facilitate the automation of customer response to various demand/response programs and dynamic pricing, through a communicating client.

Such an approach fits with the DOLFIN way, and the Smart Grid Controller will act as a gateway which will receive Demand/Response Events from a Demand Response Automation Server and will convert them into information for the Energy Efficiency Policy Maker and Actuator.

3 Interactions

This section describes the interactions of the Smart Grid Controller.

3.1 Demand/Response Event Information

Demand/Response Events are typically designed to carry a variety of information to cause reactions by customers of a utility's request. In some cases prices are used to trigger responses by a customer to the DR events, while in other cases it might be a shift level event. In general there can be a wide range of different types of information associated with a DR event depending upon how the DR strategy of a utility is designed. This information is represented by *EventInfoInstance* entities. Example data held by one of these Demand/Response *EventInfoInstance* entities includes:

- *name* – This is the name of the type. Analogous to a variable name.
- *typeID* – This identifies the type of information and may take on one of the following values:
    - PRICE_ABSOLUTE-Price number, e.g. €0.25
    - PRICE_RELATIVE-Change in price, e.g. +- €0.05
    - PRICE_MULTIPLE-Multiple of current price, e.g. 1.5
    - LOAD_LEVEL-Amount of load based on an enumeration, e.g. moderate, high, etc.
    - LOAD_AMOUNT-Fixed amount of load to shed or shift, e.g. 5MW
    - LOAD_PERCENTAGE-Percentage of load to shed or shift, e.g. 10%
    - GRID_RELIABILITY-Number signifying the reliability of the grid
- *scheduleType* – This specifies how a schedule may be associated with the DR Event information is defined and may take on the following values:
    - NONE–There is no schedule and thus the EventInfoInstance does not change values during the entire DR event ACTIVE state.
    - DYNAMIC–The time schedule is not fixed during configuration, but can be set when the DR event is issued.
    - STATIC–The schedule is fixed when the DR program is configured within the DRAS
- *Schedule* – If the scheduleType is STATIC then this is the configured schedule. A schedule is a sequence of time slots that are valid over the entire ACTIVE period of a DR event. Each time slot may take on a different value in the *EventInfoInstance*

- *enumerations* – This is a list that defines a fixed set of values that the EventInfo instance may take. If defined, the *EventInfoInstance* is an enumeration and can take on any of the values in the list. If left undefined, the EventInfo instance can take on any contiguous value between the *minValue* and *maxValue*.
- *minValue* – Minimum possible value of an *EventInfoInstance*.
- *maxValue* – Maximum possible value of an *EventInfoInstance*.

These Demand/Response Events, after going through the Smart Grid Controller and being passed to the Energy Efficiency Policy Maker and Actuator, may affect how DOLFIN places its VMs.

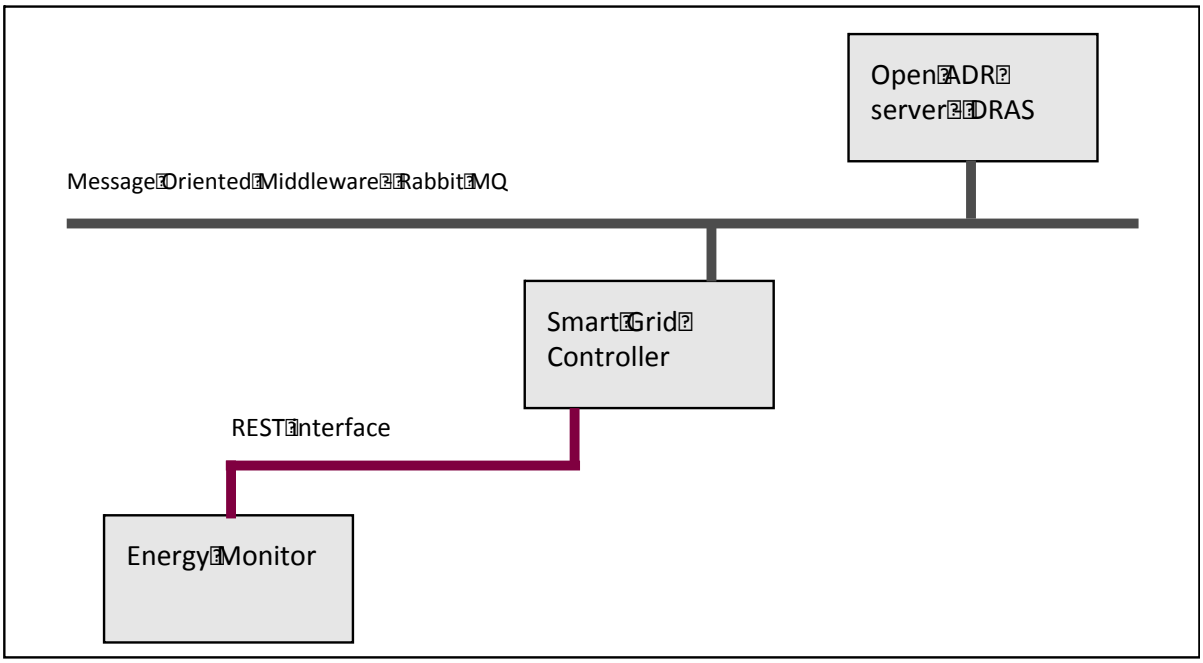

Figure 2: Smart Grid- DOLFIN communication architecture

4 Implementation

4.1 Automated Demand Response Server

Open Automated Demand Response (OpenADR) provides a non-proprietary, open standardized Demand Response (DR) interface that allows electricity providers to communicate DR signals directly to existing customers using a common language over an existing communication channel such as the Internet [1], and which employs the Automated Demand-Response (ADR) information exchange model.

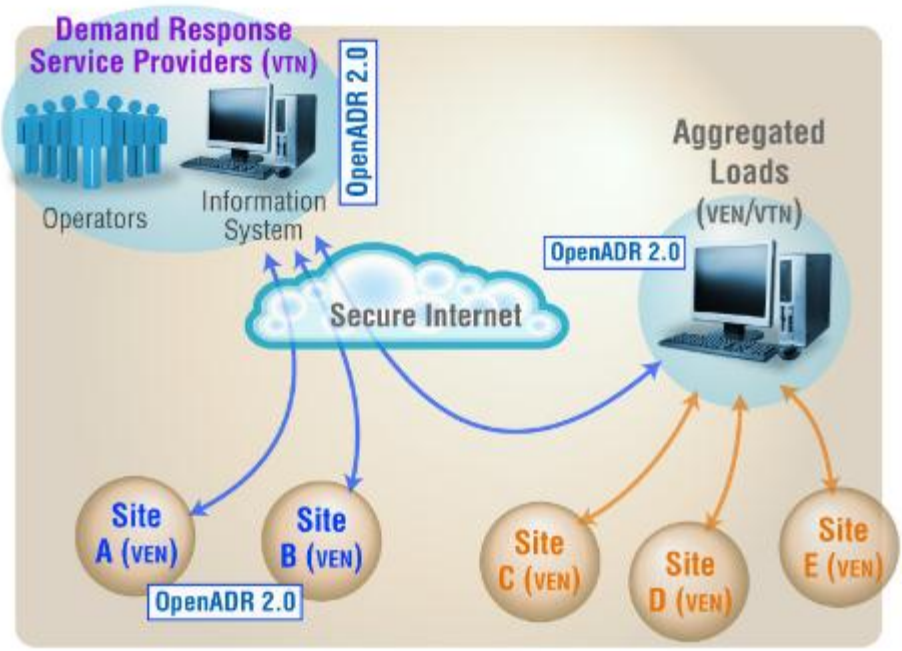

Figure 3: OpenADR communication model

OpenADR consists of two groups of nodes, as depicted in Figure 3: (I) Virtual Top Nodes (VTNs), which publish information about events to other nodes (e.g., utilities), and (II) Virtual End Nodes (VENs), which receive and process the information from the VTNs (e.g., end users).

Both VTNs and the Automated Demand Response (ADR) Server (the VTN involved in our project) are responsible for communicating grid conditions (e.g., prices, reliability events, loads) to other entities (i.e., VENs) that control demand side resources. The VTN is able to communicate with both the Grid and the VEN devices or systems in its domain. A VTN may take the role of a VEN interacting with another VTN.

A VEN has operational control of a set of resources and/or processes and is able to control the electrical energy demand of these in response to an understood set of Smart Grid messages (i.e., DR

signals). The VEN is able to communicate with a VTN receiving and transmitting Smart Grid messages that relay grid situations, conditions, or events, as well as to take the role of a VTN in other interactions, if necessary [2].

Usual examples of a VEN are thermostats, water heaters, as well as more complex systems, i.e. for smart appliance or for energy management (see Figure 4).

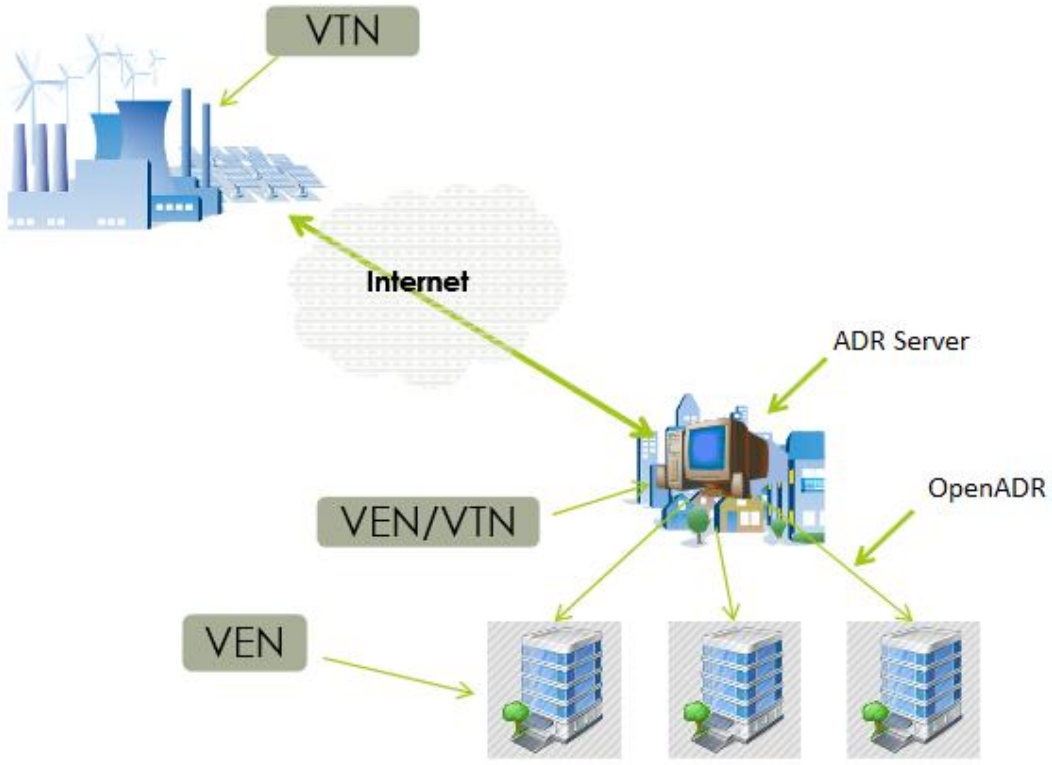

Figure 4: Commercial & industrial connection between ADR Server and end-users

As already explained, for the communication between the nodes various protocols can be used. DOLFIN has decided in our specific case to use the RabbitMQ protocol.

4.2 Message Oriented Middleware - RabbitMQ

RabbitMQ is a message oriented middleware which is a message broker system which allows a process, *the publisher*, to write messages to an exchange, and these messages are consumed by another process, called the *consumer* as depicted in Figure 5. Figure 5 also depicts the two main internal components of the message broker: (I) the exchange, which represents the entry point; and the (II) queue component, which acts like a container in which relevant messages are stored until their consumption.

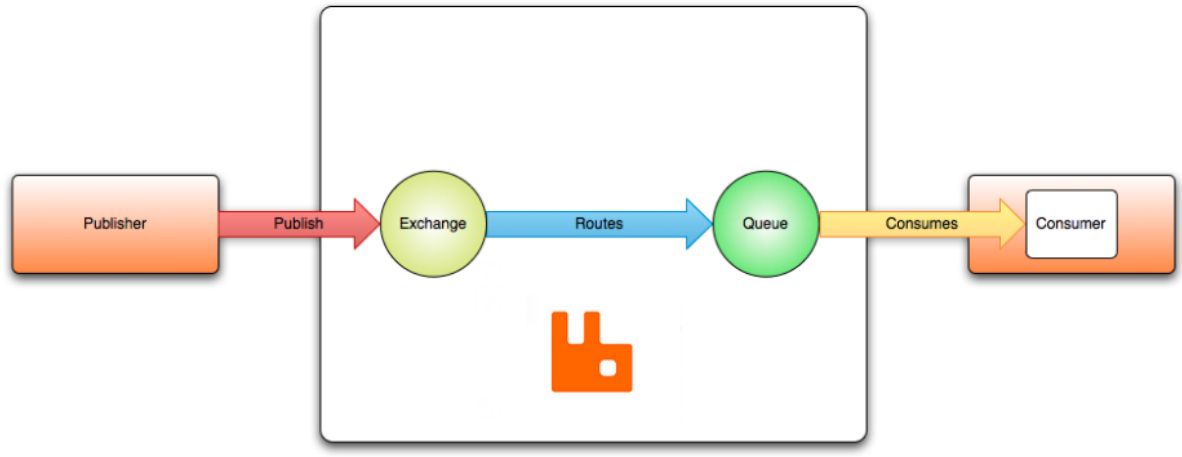

Figure 5: Direct message exchange in RabbitMQ

The main advantages of using RabbitMQ, as listed in [3], are:

- **Reliability** - RabbitMQ offers a variety of features to trade off performance with reliability, including persistence, delivery acknowledgements, publisher confirms, and high availability.
- **Flexible Routing** - messages are routed through exchanges before arriving at queues. RabbitMQ features several built-in exchange types for typical routing logic. For more complex routing we can bind exchanges together or even write our own exchange type as a plug-in.
- **Highly Available Queues** - queues can be mirrored across several machines in a cluster, ensuring that even in the event of hardware failure your messages are safe.
- **Multi-protocol** - RabbitMQ supports messaging over a variety of messaging protocols.
- **Support for programming languages** - the clients for RabbitMQ broker can be implemented using several programming language, including JAVA, Ruby, Python, .NET, C/C++

These are all features which benefit DOLFIN.

4.3 Smart Grid Controller

The Smart Grid Controller will accept the ADR Server messages and build an internal calendar of prices for each time slot in a day. The messages are defined in section 4.2.

After receiving a PRICE_ABSOLUTE message the Smart Grid Controller will build an internal calendar based on the *startDate* and *endDate* of the message and the pricing data held in the schedule of the message. The schedule contains different time intervals with different prices during a day.

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Sun	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Mon	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Tue	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Wed	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Thu	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Fri	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Sat	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2

Table 1: Internal calendar of prices in hourly slots after receiving the PRICE_ABSOLUTE message

After the receipt of the PRICE_RELATIVE message, which represents Monday 7th September between 18:00 and 20:00 the pricing calendar will be updated. Messages of type PRICE_RELATIVE cause the calendar price to have the price in the message added to the value. In this case 0.4 will be added to the base price of 0.2.

The resulting calendar will look like:

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Sun	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Mon	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.2	0.2	0.2	0.2
Tue	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Wed	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Thu	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Fri	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2
Sat	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2

Table 2: Internal calendar showing PRICE_RELATIVE adjustment

The Smart Grid Controller will send messages to the Energy Monitor and all other interested components with the current price of the electricity. This message will be send using the Dolfin format.

4.4 System Overview

The interaction of the ADR Server with the Smart Grid Controller (SGC) is depicted in Figure 6: both Smart Grid and Energy Provider (EP) messages, dispatched via RabbitMQ, are sent by the ADR Server to be processed by the Smart Grid Controller (SGC).

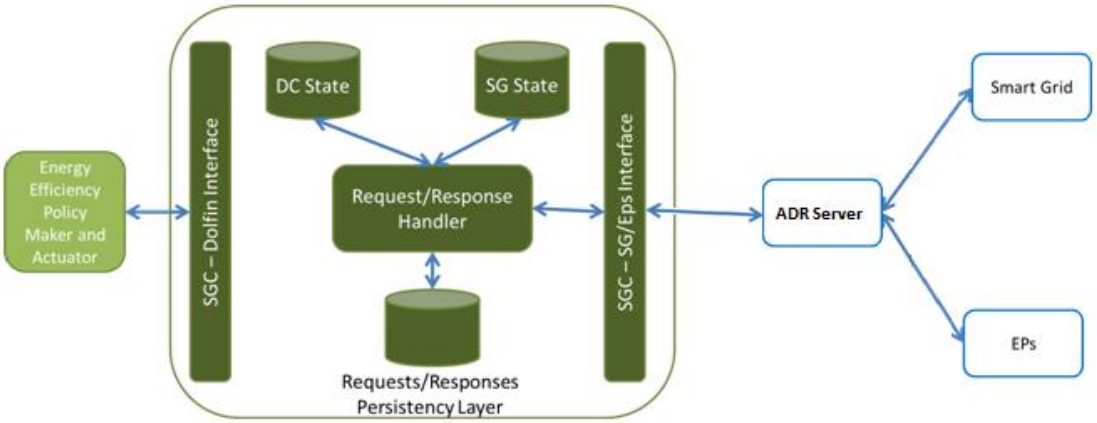

Figure 6: The ADR Server interaction with Smart Grid Controller

The main scope of the message exchanged between ADR Server and Smart Grid Controller is to convey information related to:

- Price / Load information
    - Dynamic price signals
    - Reliability signals
    - Load predictability and generation information
    - Emergency signals
- Demand-Response schedule.

This exchange facilitates enterprise interaction with energy markets, which:

- Allows effective response to emergency and reliability events,
- Allows taking advantage of lower energy costs by deferring or accelerating usage,
- Enables trading of curtailment and generation,
- Supports symmetry of interaction between providers and consumers of energy,
- Provides for aggregation of provision, curtailment, and use [5].

5 Implementation description

The OpenADR standard is using an established message structure, selected depending on the message purpose. In the 2.0 version of the standard, several features are available for customers:

- **Quote or Dynamic Prices (EiQuote)** – to distribute complex dynamic prices such as block and tier tariff communication but just as an information
- **Reporting or Feedback (EiReport)** – used to set the state of a resource (response)
- **Slow DR** - Demand Response where the signals are sent significantly before the events are called, such as day-ahead.
- **Event (*EiEvent*)** - the core DR event functions and information models for price responsive DR. This service is called to perform a transaction. The service parameters and event information distinguish different types of events. Event types include reliability events, emergency events, and more – and events may be defined for other actions under a transaction [2].

An *EiEvent* includes information regarding to:

- *name* – Event description
- *typeID* – This identifies the type of information exchanged and for now may be:
    - PRICE_ABSOLUTE (Hourly prices)
    - PRICE_RELATIVE (Energy price change)
    - PRICE_MULTIPLE (Energy price multiplier)
- *scheduleType* – This specifies if the *EiEvent* will change customer's schedule and it may take one of the following values:
    - NONE – The *EiEvent* does not change customer's schedule values during the event
    - DYNAMIC – The schedule is not fixed during configuration and may be changed by events
    - STATIC–The schedule is fixed when the DR program is configured
- *Schedule* – Represents a list of schedule events that will occur including information about the time slot and the correspondent value (e.g., price, multiplier)
- *Priority* – Specifies the event priority (e.g., High – for events that can change the schedule, Normal, Low – for notification events)

- *ResponseRequired*
- *Timestamp*
- *StartDate*
- *EndDate*

The *EiEvent* data is mapped into the data structure described in the following section.

5.1 Message Structure

For the DOLFIN project [4], we agreed on using the *EiEvent* type messages, since they allow us to receive all the information needed by the later stages of the processing, in which the optimization process is performed. The structure of such an *EiEvent* is described in Snippet 1:

```
public class EiEvent {
	String name;
	String typeID;
	String scheduleType;
	List <ScheduleEvent> schedule;
	String priority;
	Boolean responseRequired;
	long timestamp;
	String startDate;
	String endDate;

}
```

Snippet 1: *EiEvent* message structure

The supporting structures, ScheduleEvent and Interval are shown below:

```
public class ScheduleEvent {
	Interval interval;
	double value;
}
```

Snippet 1a: ScheduleEvent message structure

```
public class Interval {
	String start;
	String end;
}
```

Snippet 1b: Interval message structure

5.2 Scheduling

5.2.1 Hourly prices schedule (PRICE_ABSOLUTE)

A price absolute event represents a price type message that contains the hourly prices valid throughout an established time interval (i.e., day, week, month). The message containing the event is generated and distributed to the clients a day before the absolute prices changes are enforced.

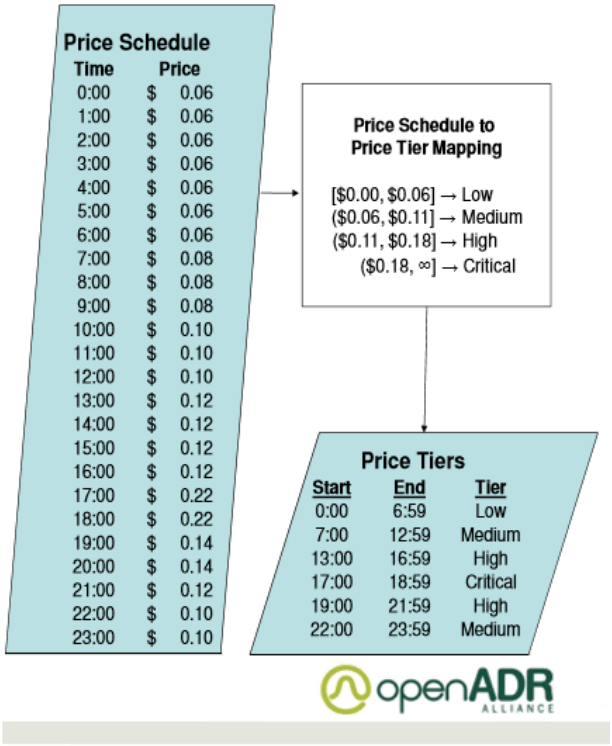

Figure 7: Hourly energy price

The Absolute price message is sent in JSON format and contains the following information:

```
{"name":"Hourly energy prices [€/kW]",
"typeID":"PRICE_ABSOLUTE",
"scheduleType":"STATIC",
{
"schedule":[{
      "interval":{
            "start":"00:00",
            "end":"00:59"},
      "price":0.06},
      {
      "interval":{
            "start":"01:00",
            "end":"01:59"},
```

```
            "price":0.06},
            {
            "interval":{
                  "start":"02:00",
                  "end":"02:59"},
            "price":0.06},
            …
            {
            "interval":{
                  "start":"22:00",
                  "end":"22:59"},
            "price":0.1},
            {
            "interval":{
                  "start":"23:00",
                  "end":"23:59"},
            "price":0.1}],
      "priority":"NORMAL",
      "responseRequired":false,
      "timestamp":1441014310932,
      "startDate":"2015-09-20 00:00",
      "endDate":"2015-09-29 23:59"}
```


On receiving the message, each customer is updating the prices per hour in his internal prices schedule.

5.2.2 Energy price change (PRICE_RELATIVE)

If during a day or a time period the price is modified, an "Energy price change" is sent to customers containing the time interval and the new energy price for that interval. Example data held by the price_relative event includes:


```
      {"name":"Energy price change [€/kW]",
      "typeID":"PRICE_RELATIVE",
      "scheduleType":"DYNAMIC",
      "schedule":[{
            "interval":{
                  "start":18.0,
                  "end":19.59},
            "price":0.4}],
```

```
"priority":"HIGH",
"responseRequired":false,
"timestamp":1441019816749,
"startDate":"2015-09-20 18:00",
"endDate":"2015-09-20 19:59"}
```


5.2.3 Energy price multiplier (PRICE_MULTIPLE)

When the exact energy price is unknown, the energy provider can use a multiplier value applied to the current tariff for a certain time. In this case the ADR Server will send an "Energy price multiplier" message containing the time period and the multiplier value.

Price Multiplier (Current Price Multiplier field)	Price ($/kW)
1 or if there is no event and if no information is provided in the current price multiplier variable	Base price $0.10
2	$0.20 and < $0.40
4	$0.40 or more

Table 3: Price multiplier mapping


```
{"name":"Energy price multiplier [€/kW]",
"typeID":"PRICE_MULTIPLE",
"scheduleType":"DYNAMIC",
"schedule":[{
      "interval":{
            "start":18.0,
            "end":19.59},
      "multiplier":2}],
"priority":"HIGH",
"responseRequired":false,
"timestamp":1441019952416,
"startDate":"2015-09-20 18:00",
"endDate":"2015-09-20 19:59"}
```

6 Integration & Implementation description

In order to validate the correctness of the Smart Grid Controller's operation as part of DOLFIN, the module is to be integrated in stages, testing parts of its functionality in each stage. Given that the internal architecture of each module is designed so as to make its primary function as efficient as possible, a REST API was defined for the interaction between different DOLFIN components. In this section, we define the API of the Smart Grid Controller exposed to other DOLFIN modules and provide a description of the workflows through which the system accesses SGC functionality.

6.1 Integration design

In this section we present the general design of the integration layer responsible for communicating interactions between the Smart Grid and DOLFIN to different DOLFIN modules. In essence, the SGC exchanges information only with the Energy Efficiency Policy Maker and Actuator as seen in Figure 6. Also, it is a fact that most of the information is one-way, that is it flows from the ADR through the SGC to the Policy Maker and short responses are periodically communicated to the Smart Grid, usually for high priority events or requests from the EP or the Grid.

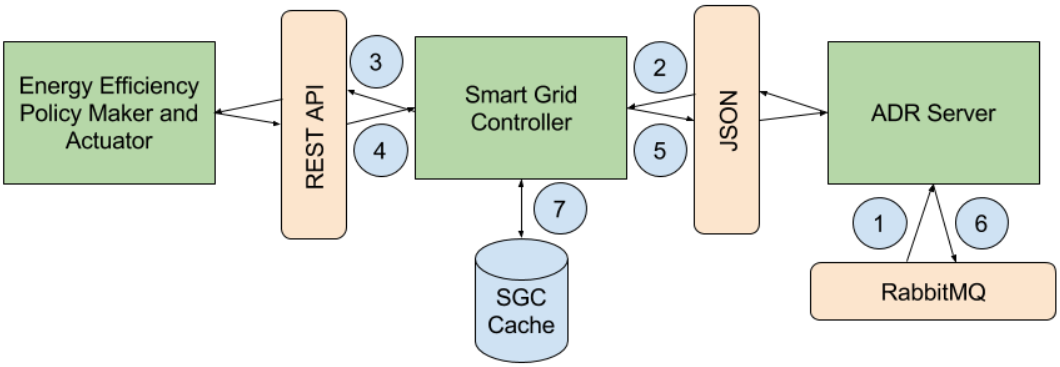

Figure 8: Data flow between modules

In Figure 8 the flow of information through the different components is depicted in stages. These are:

1. Messages on the EP/Smart Grid queues are received using the RabbitMQ protocol. These messages conform to the OpenADR specification and their format is described in detail in Section 2.1.

2. The ADR server aggregates and pushes significant events to the Smart Grid Controller using JSON messages, as described in Section 4.

3. The SGC exposes new messages to the Policy Maker through the SGC REST API, which is periodically polled by the Policy Make for minor changes while high priority messages are pushed immediately.

4. The Policy Maker can request information on the energy price schedule periodically, or inform the Smart Grid Controller on appropriate responses to be propagated through the ADR.

5. The SGC provides appropriate responses to ADR requests, as per the instructions of the Policy Maker.

6. The ADR responds to the Smart Grid through RabbitMQ.

7. Specific data that are needed to minimize the SGC data transfer and provide the Smart Grid or DC state to requesting modules is locally cached. As described in Section 1.2, the information cached includes:

    a. The current and near future computing load of the DC

    b. On-site energy generation or re-use details of the DC

    c. Data relating to the current energy tariffs or their trend

    d. The availability of energy in the Smart Grid along with grid stability metrics

6.2 Smart Grid Controller REST API

Given that the internal communication protocol with the Smart Grid is RabbitMQ, message translation needs to be performed to transform messages to the REST API used by DOLFIN. Such translation is non-trivial and would include preliminary sorting, verification and filtering of the raw Smart Grid messages received by SGC on the queue. Thus the SGC provides a RESTful service that can be polled to extract information and send high-level messages to the SGC, which are then translated and communicated using the RabbitMQ protocol. The Smart Grid Controller will send messages to the Energy Monitor and all other interested components with the current price of the electricity. This message will be send using the Dolfin format.

```
{
      message: “energy_price”
      timestamp: 1441019952416
      payload: {
            type: PRICE_ABSOLUTE
            price: 0.3
      }
}
```

7 Conclusions

This document has presented the design and implementation attributes of the Smart Grid Controller. The Smart Grid Controller acts as the component that integrates the DOLFIN platform with the Smart Grid network. The purpose of this component is to be able to interact with the energy related requests from the Smart Grid side in order that DOLFIN can initiate its Energy Management cycle when the relevant conditions arise.

As described earlier, the Smart Grid network issues a request, towards the DC, for lowering the power consumption. We have seen that this expressed as messages from the ADR Server with:

- an absolute figure, determining the exact level of power consumption that should be followed, as seen in the PRICE_ABSOLUTE messages
- a percentage of decrease to the current levels of power consumption, as seen in the PRICE_RELATIVE and PRICE_MULTIPLE

The role of the Smart Grid Controller is to intercept and perform an initial examination of such message requests from the Smart Grid.

The Smart Grid Controller is required to exchange messages with the Smart Grid network and be able to parse them, so as to create an internal representation of the message that would be understandable by the DOLFIN platform. These ADR server messages are processed and then manifested in the calendar structure held by the Smart Grid controller.

The Smart Grid Controller component is written in a way that it incorporates a capability to communicate with possibly different protocols with the Smart Grid, thus acting as a kind of gateway for interfacing with various Smart Grid networks, although the current implementation is focussed on the ADR messages.

8 References

1. Barry Haaser, "Enabling The Standard for Automated Demand Response" http://www.openadr.org/assets/docs/openadr%20plma%20seminar_final.pdf

2. Ulrich Herberg, Jim Zuber - "OpenADR 2.0b Profile Specification – B profile" 07 January 2013, Morgan Hill, Document Number: 20120912-1

3. "RabbitMQ", [Online] Available: https://www.rabbitmq.com

4. DOLFIN deliverable D4.1: "Synergetic Data Centres for energy efficiency" – May 2015, www.dolfin-fp7.eu

5. James Mater - QualityLogic, "OpenADR Technology Overview", 28 October 2013, Atlanta, http://c.ymcdn.com/sites/www.peakload.org/resource/resmgr/2013fallarchive/2_openadr_plma_tech_overview.pdf