Facilitating a generic communication interface to distributed energy resources

Mapping IEC 61850 to RESTful services

Pedersen, Anders Bro; Hauksson, Einar Bragi; Andersen, Peter Bach; Poulsen, Bjarne; Træholt, Chresten; Gantenbein, Dieter

Published in:
First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010

Link to article, DOI:
10.1109/SMARTGRID.2010.5622020

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Pedersen, A. B., Hauksson, E. B., Andersen, P. B., Poulsen, B., Træholt, C., & Gantenbein, D. (2010). Facilitating a generic communication interface to distributed energy resources: Mapping IEC 61850 to RESTful services. In First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010 IEEE. DOI: 10.1109/SMARTGRID.2010.5622020

Facilitating a Generic Communication Interface to Distributed Energy Resources

Mapping IEC 61850 to RESTful Services

Anders Bro Pedersen
Technical University of Denmark

Einar Bragi Hauksson
Technical University of Denmark

Peter Bach Andersen
Technical University of Denmark

Bjarne Poulsen
Technical University of Denmark

Chresten Træholt
Technical University of Denmark

Dieter Gantenbein
IBM Research, Zurich

***Abstract*—As the power system evolves into a smarter and more flexible state, so must the communication technologies that support it. A key requirement for facilitating the distributed production of future grids is that communication and information are standardized to ensure interoperability. The IEC 61850 standard, which was originally aimed at substation automation, has been expanded to cover the monitoring and control of Distributed Energy Resources (DERs). By having a consistent and well-defined data model the standard enables a DER aggregator, such as a Virtual Power Plant (VPP), in communicating with a broad array of DERs. If the data model of IEC 61850 is combined with a set of contemporary web protocols, it can result in a major shift in how DERs can be accessed and coordinated. This paper describes how IEC 61850 can benefit from the REpresentational State Transfer (REST) service concept and how a server using these technologies can be used to interface with DERs as diverse as Electric Vehicles (EVs) and micro Combined Heat and Power (µCHP) units.**

I. Introduction

Several standards are currently under development that can facilitate communication with DERs in a smart grid constellation. Among these are OpenARD [1], OpenAMD [2] and EMIX [3] as well as the IEC 61850 standard, on which this paper will focus. The standard is thoroughly researched and has been described in several papers [4] [5]. The standard offers a structural decomposition of the units to which it communicates. This means that each subcomponent of a DER can be described by the information model of IEC 61850. This makes the standard suitable for scenarios in which an aggregator needs a fine-grained knowledge and control of the state, structure and operation of a DER. The standard is a good match for the virtual power plant concept, in which an intermediate entity represents an aggregated group of DERs in the power system and on the market. Apart from the syntactic level, the standard also defines the protocols that carry the data over a network. Without providing any recommendations on the medium used, the standard proposes the use of the TCP/IP protocols to enable internet communication. On the upper part of the ISO OSI stack, the standard describes the use of the Manufacturing Message Specification (MMS) standard as an application-level protocol [6]. Replacing MMS with REST services will, however, have certain advantages. A REST service is a special flavor of web services which are connected to the concept of a Service Oriented Architecture (SOA). As REST services represent both a simple and well- documented concept for achieving a high degree of interoperability, they are a good candidate for use in the IEC 61850 protocol stack. The next chapter describes IEC 61850 and REST in greater detail. Then, the paper describes how an IEC 61850 server was developed based on the above technologies, and finally a case is defined for interfacing with a µCHP unit and an electric vehicle. The IEC 61850 via REST implementation suggested in this paper is used in the Danish EDISON project [7], where a VPP-like aggregator should coordinate the charging of electric vehicles, as well as in the Danish Generic Virtual Power Plant project, where µCHPs are coordinated to support the grid.

II. The IEC 61850 Standard

The IEC-61850 standard was designed to enable interoperability between different devices in the substation environment, and to facilitate the adaptation of future networking technologies. One of the cornerstones of the standard is a layered data model (see Figure 1), which has been designed to closely model the physical substation environment.

A. Data model

The logical device is a virtual representation of a physical device within the substation. It is comprised of a name, a path to the object itself and a list of logical nodes. A logical device contains one or more logical nodes, which represent various components in the physical device.

As an example, the setup tested contains a logical node called *MMXN1*. The *MMXN* part of the name refers to the type of the node, which in this case is *"Non-phase-related measurements"* [8]. Apart from the name, the MMXN class must also contain all the data classes defined by the *Logical Node Class* defined by the standard [8]. Beside this, instances of the *MMXN* class can include a list of optional data classes and attributes. The data classes represent meaningful information inside the nodes and are declared recursively.

An attribute can either be a simple type, such as *FLOAT32*, *INT24* or *BOOLEAN*, or it can consist of both other simple and complex attributes.

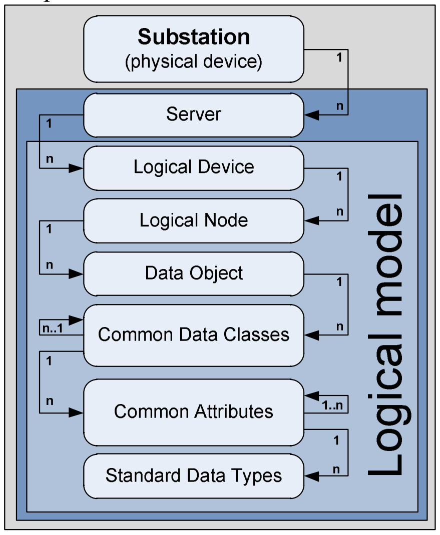


Figure 1 – IEC 61850 data model hierarchy [9]

B. Data-Sets

The logical nodes can contain data-sets, which contain sets of data that have a natural association [10]. Data-sets are primarily used for reporting and logging [10], but can also be requested directly. Included in the data-sets are properties stating when a report/log should be triggered. An example of a triggering condition is “Data Change”.

C. Logging and reporting

The IEC 61850 outlines a reporting mechanism, which is essentially a payload-carrying event that is sent back to the subscribing clients when triggered. Reporting is directly linked to data-sets as it is the data attributes in the data-sets that specify the trigger conditions. Closely related to reporting is logging. They both rely on the data-sets for triggering, but the reports are sent back to the clients, whereas the logs are persisted locally [10].

D. Object references

Any object within the data model can be referred to directly via its object path [10]. Because of the tree-like properties of the model, this path resembles a fully-qualified file-name notation. The path lists all the objects on the route from the root of the model to the object in question. Where fully-qualified file-name notation usually has a fixed delimiter between object names, the IEC-61850 references use a slash to separate the logical device from the rest of the path, which is then separated by periods.

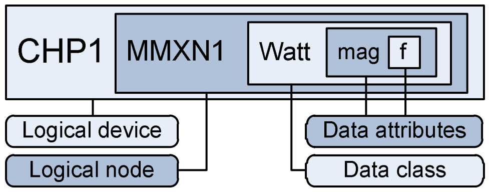


Figure 2 – Illustration of an IEC 61850 reference [9]

Included in the object reference is a filtering mechanism called *functional constraints* (FC), which is used to group the data-attributes. The functional constraints are usually specified at the end of an object reference, incased in square brackets. For the DC example, this would result in the path:

CHP1/MMXN1.Watt.mag.f[DC]

Because the paths resemble a fully-qualified file-name notation, they can easily be mapped to a URL, which makes the data model near perfect for REST. As the physical device and the server are not referred in the object path, a REST URL could look something like:

http://hostname/device/node/class/attribute

Included in IEC 61850 is also a service specification called the *Abstract Communication Service Interface (ACSI)* [11], which outlines a set of methods that are used when communicating with the system. These methods have been mapped to REST.

III. REST SERVICES

The REST architectural style was first described in 2000 in a Ph.D. dissertation by Roy Fielding [12]. The industry did, however, not embrace REST right away, probably because at about the same time, the *Single Object Access Protocol* (SOAP) [13] was embraced by most of the large software vendors and therefore got a lot of attention. As time passed and SOAP grew, adding numerous extensions, people started looking for a lighter, more web-centric alternative. Today many big web companies provide REST services for others to interact with their systems. These include companies such as Amazon, Google, Yahoo and Facebook.

Where SOAP comes complete with a seemingly ever-growing suite of extensions, the primary aim of REST is to stick closer to the basic functionality of the HTTP protocol on which the web is based. Unlike SOAP, there is no standard available that describes REST. Instead, developers should follow the REST principles when creating a *RESTful* [14] service.

The most important REST principle is to expose the resources in a RESTful service as unique URLs. An example of this might be a library service where the whole book catalog could be accessed by using the URL http://library.com/books/ whereas a single book could be accessed using the URL http://library.com/books/123 (where 123 is the ID of the book). As opposed to SOAP services, where you would issue a method call such as *createBook, deleteBook,* RESTful services utilize the

HTTP methods, GET, POST, PUT and DELETE for reading, creating, updating and deleting resources, respectively.

The REST principles do not define any specific format for request or response data. Most common, however, is the XML format, but other formats such as *JavaScript Object Notation* (JSON) [15] are becoming increasingly popular, especially in *Asynchronous JavaScript And XML* (AJAX) services. Ideally the client is able to ask for specific representation by setting the *Accept* request header to the desired format name.

Like the HTTP protocol itself, RESTful services are stateless, meaning that no state should be stored on the server between requests from the client. Each request should therefore contain all the information necessary to serve the client. RESTful services also embrace other aspects of the HTTP protocol, such as *status codes*, *conditional get* and *caching*.

Status codes are sent along with any HTTP response and indicate the type of the response. Examples of common status codes are 200 (OK) for a successful request and 404 (NOT FOUND) for a request for a non-existing resource. HTTP defines status codes in the range 1xx to 5xx.

Conditional Get allows the client to ask the server whether the requested resource has changed since it was last retrieved. This is achieved either by sending the *If-Modified-Since* header with its value set to the time of the last retrieval or by sending the *eTag* header that came with the last response. If the resource has not been modified the server only returns status code 304 (NOT MODIFIED). This can be important for performance, as unchanged data does not need to be re-sent.

Another closely related header is the cache header, sent along with the response from the server. The cache header allows the server to tell the client whether and for how long the client should cache the response. This improves performance as it saves the client from requesting the same data again, if it is not expected to change. This is mainly used for static resources, such as images on the web, but can also be utilized in RESTful services.

The functionality of RESTful services is very closely related to the functionality of the web, which has been extremely successful over the last couple of decades. This is mainly due to its scalability, interoperability and the fact that it is simple and easy to understand. HTTP and the WWW are used for a wide variety of tasks ranging from personal home pages to secure internet-banking and e-commerce. HTTP already has a built-in authentication mechanism but since many, if not all of the tasks mentioned call for strict security, several protocols have developed to add things like encryption to HTTP. Among some of the more well-known ones are open standards, such as the TSL/SSL [16] protocols. Because of technologies like these and the needs from which they arose, the idea of using REST for communication in electrical systems, as is the case with the VPP and the distributed energy resources, could surely be considered safe.

IV. RESTful interface for IEC 61850

As mentioned, the IEC 61850 reference paths resemble a fully-qualified file-name notation or a URL. This simplifies the task of creating a RESTful interface for the IEC 61850 data model. The idea is that the various objects in the data model hierarchy can be thought of as resources, which can be accessed by using the IEC 61850 object reference. As an example the data attribute

CHP1/MMXN1.Watt.mag.f

belonging to logical device "CHP1", logical node "MMXN1" and so on would have the URL

CHP1/MMXN1/Watt/mag/f

and the entire "Watt" data object could be retrieved with the URL *CHP1/MMXN1/Watt.* For getting data, as in the example above, an HTTP GET request would be used. The server would then respond with the requested data in XML format as shown in Figure 3. For setting data, the same URL would be used, but with the HTTP method POST instead of GET.

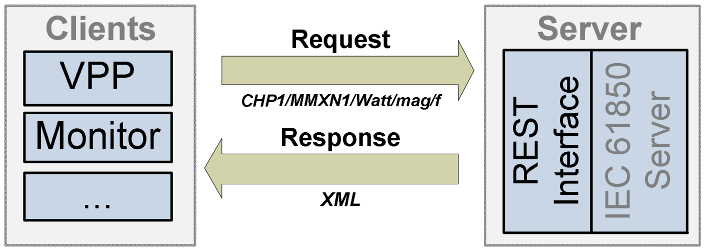


Figure 3 - REST communication overview.

The ACSI interface described in IEC 61850-7-2 enables clients to inspect the data model, to read and write data, and to access data-sets, logs and more. To do so, the client calls methods such as *GetDataValues* and *SetDataValues*. These methods resemble traditional methods in a programming language with input arguments and return values. As described above, the REST architecture guidelines define services as a set of resources instead of methods as in the case of ACSI.

To make a resource-oriented interface for the IEC 61850 standard, a mapping from the ACSI methods to URL and HTTP method-pairs has been defined. The mapping for the data model of the IEC 61850 standard is shown in Table 1.

Mappings for data-sets, reporting, logging, setting-groups and substitutions can be made in a similar manner. These mappings are presented in full detail in [9].

The first column in Table 1 shows the URL template of a resource, the second column shows the HTTP method used to get or set the URL, and the third column shows the ACSI services this resource replaces. The response from the GET requests to the resources shown in Listing 1 can be modified by using *query string* parameters. Three query string parameters have been defined: *expandLevel*, *fc* and *includeValues*.

The *expandLevel* parameter controls how big a portion of the data model is retrieved. The default value of this parameter is zero, which results in only the requested level of the data model being retrieved, as can be seen in Listing 1. Setting the *expandLevel* parameter to a higher number or to the value “all”, allows the client to retrieve bigger portions of the data-model hierarchy, as seen in Listing 2. By retrieving the whole data model, the client can easily inspect an entire logical device and a single URL can therefore replace multiple ACSI services.

The two remaining parameters accepted by the RESTful service are *fc* and *includeValues*. The *fc* parameter is used for filtering the data-model hierarchy to include only data attributes with the specified functional constraint. The *includeValues* parameter specifies whether the data values should be included in the response in addition to the data model.

TABLE 1 - ACSI TO REST MAPPING

Url	Meth.	ACSI equivalent
/	GET	GetServerDirectory (GetAllDataValues) (GetDataValues)
/[LD]/	GET	GetLDDirectory (GetAllDataValues) (GetDataValues)
/[LD]/[LN]	GET	GetLNDirectory (GetAllDataValues) (GetDataValues)
/[LD]/[LN]/[DO]	GET	GetDataDirectory GetDataDefinition (GetAllDataValues) (GetDataValues)
/[LD]/[LN]/[DO]/[DA]	GET	(GetAllDataValues) (GetDataValues)
/[LD]/[LN]/[DO]/[DA]	POST	SetDataValues

As Figure 3 illustrates, the response format of the RESTful service is in XML. Because of XMLs ability to model arbitrary data structures it is possible to represent the hierarchical IEC 61850 data model in a concise and readable format.

To show this, two URLs and their resulting output are given. Listing 1 shows a response containing a single value; Listing 2 is an example of what a larger view of the data model might look like.

```
<DA Name="f" Type="FLOAT32"
Ref="EV1/ZCEV1/MaxRtDchPwr/setMag/f">16800</DA>
```

Listing 1 - Response for URI: /EV1/ZCEV1/MaxRtDchPwr/setMag/f

As seen in Listing 2, a request for an entire data hierarchy can be quite verbose and might not be suitable for frequent use in a production environment. This kind of request is, however, very useful during the configuration and development phase when new devices need to be discovered. As the RESTful interface uses standard HTTP GET request, it is even possible to use a web browser to discover an IEC-61850-enabled device. As with ACSI methods such as *GetServerDirectory* and *GetLDDirectory*, the RESTful interface also enables clients to programmically discover devices in a generic manner.

```
<DO Name="MaxRtDchPwr" Type="ASG"
Ref="EV1/ZCEV1/MaxRtDchPwr">
<DA FC="SP" Name="setMag" Type="Struct"
Ref="EV1/ZCEV1/MaxRtDchPwr/setMag">
<DA Name="f" Type="FLOAT32"
Ref="EV1/ZCEV1/MaxRtDchPwr/setMag/f">16800</DA>
</DA>
<DA FC="CF" Name="units" Type="Struct"
Ref="EV1/ZCEV1/MaxRtDchPwr/units">
<DA Name="SIUnit" Type="Enum"
Ref="EV1/ZCEV1/MaxRtDchPwr/units/SIUnit">Watt</DA>
<DA Name="multiplier" Type="Enum"
Ref="EV1/ZCEV1.MaxRtDchPwr/units/multiplier">Item</DA>
</DA>
<DA FC="DC" Name="dU" Type="Unicode255"
Ref="EV1/ZCEV1/MaxRtDchPwr/dU">Maximum rated discharging
power</DA>
</DO>
```

Listing 2 - Response for URI: /EV1/ZCEV1/MaxRtDchPwr/?expandLevel=all

Although XML has been chosen as the output format in the example above, the RESTful interface could just as well use other output formats. Thanks to the Accept header discussed above, the same RESTful service could both accept and output XML and JSON. One argument for using the JSON format is that it is designed for serializing common objects, lists and scalar values found in virtually all programming languages and can therefore easily be deserialized by those languages. And without the verbose declarations, it is definitely more compact. JSON libraries exist for all major programming languages/frameworks, including Java, .Net, C++, Python etc [17].

As discussed in Section II, the IEC 61850 standard defines a reporting mechanism. Because of the connection-less nature of HTTP, and thereby of REST, implementing reporting callbacks to the client is not entirely straightforward. However, such mechanisms exist, for example the WebHooks model [18] proposed by Jeff Lindsay, NASA Ames Research Center. The approach is to have the IEC 61850 server enable clients to subscribe to reporting events, similar to the approach described in IEC 61400-25-3[19], and pass along with that subscription request a callback URL which the IEC server can use for sending reports to the client as HTTP POST requests. Security must be kept in mind using this approach, because the client must be able to validate that the callbacks received do indeed come from the server. This could be solved using HTTP-basic authentication or X.509 client certificates [20].

V. CASE STUDIES

To test the usefulness of REST for mapping to the IEC 61850 standard, a server complying with both of these has been implemented. The server was designed "from scratch" to support multiple devices and to test the scalability of the REST implementation. The server has been written entirely in C# for the .NET framework.

To facilitate the deployment to embedded devices and add support for non-Windows systems, a small open-source in-process web server was chosen to host the REST interface. Except for the logging and reporting mechanisms, which run in the background, the server waits for requests from potential clients. When a request is received for a given URL, the server looks up the requested object in its internal representation of the data model. If the request is a "write" the server will simply return a status code as confirmation. In the case of a "read", however, the requested object is serialized to XML and returned to the client. One of the benefits of this approach is that it is relatively simple to return not just the requested object but the entire sub-model. In fact, if a request is received for the logical device the server is able to return the entire data model, including all the values, using a single response.

The goal of these case studies is to test the REST mapping against different DERs, namely,

- to describe the μCHP setup and briefly show how the communication has been mapped
- to touch upon the challenges of mapping the electric vehicles and what was needed to accomplish this.

A. *Case study: Interfacing with a μCHP*

The server was designed using a modular plug-in architecture with a generic interface, which enables easier adaptation and installation of virtually any type of device.

Among the IEC-61850-enabled devices is a pair of Dachs μCHP units from Senertec. The Dachs accepts a series of different commands over the RS232 line and each of these commands returns a different set of values. These values include electrical and temperature measurements. When a complex REST request is received, i.e., more than one data attribute is requested, the IEC 61850 server requests each required value from the Dachs module. The Dachs module then locates the requested value in the set of values returned by each of its commands. As the values are requested from the Dachs module one at a time, the module caches the result of each command for a fraction of a second to avoid having to issue a command multiple times for each REST request.

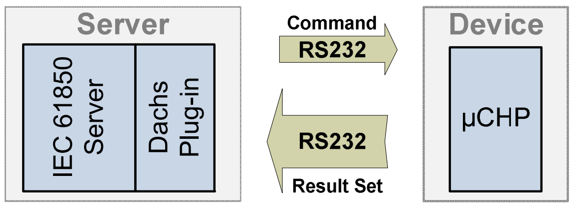
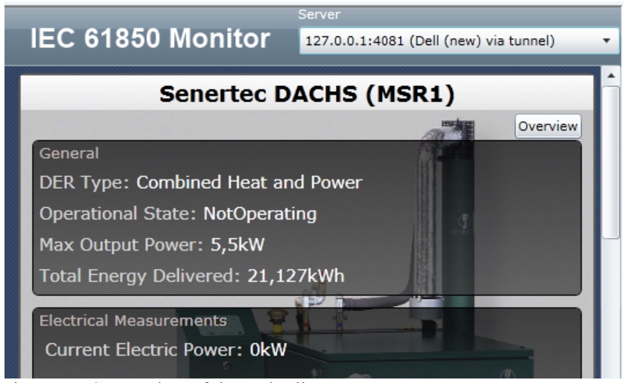


Figure 4 - μCHP module RS232 communication.

Two client applications were developed to demonstrate the system and to facilitate testing. One of these demo applications is a web client showing the various devices attached to the server, as well as all the measurements and controls available. A screenshot from the client showing one of the μCHPs can be seen in Figure 5.



Senertec DACHS (MSR1)

General

DER Type: Combined Heat and Power

Operational State: NotOperating

Max Output Power: 5,5kW

Total Energy Delivered: 21,127kWh

Electrical Measurements

Current Electric Power: 0kW

Figure 5 - Screenshot of the web client.

B. *Case study: Interfacing with an EV*

Another example of a plug-in made for the server is for an EV charging-spot. As no real vehicles were available at the time of development, the plug-in was mapped to an EV simulation instead, which further illustrates the versatility of the system. The simulation includes both a charging-spot and an EV, which contains a simulated battery. When the EV is plugged into the charging spot an EV aggregator-server can tell the charging spot to control the charging using schedules sent via the IEC 61850 REST interface. Such an EV aggregator-server is used in the EDISON project [21].

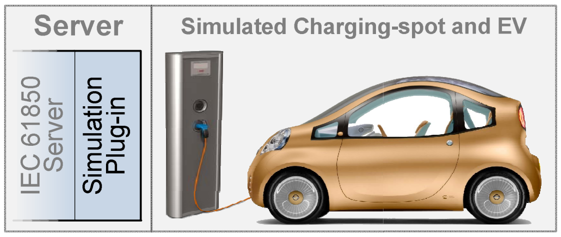


Figure 6 - Setup showing simulated EV and charging-spot.

The charging spot therefore acts as a proxy for the EV. This is in accordance to [9] and [21] and means that the EV does not need a wireless connection for being utilized by the aggregator server. Because of this, even though the charging spot and the vehicle are simulated as individual entities, they are mapped with the charging spot as the logical device and the currently connected EV represented as a logical node. The aggregator server, or any other client, can read various values from the charging-spot and the EV via the IEC 61850 interface, such as the current power usage, the phases in use and the current state of charge.

There is an addition to the IEC 61850 standard, the IEC 61850-7-420 [22], that deals specifically with DERs. After having analyzed the requirements for the charging spot and EV, it became apparent that this standard needed further extension. Logical nodes for charging-spots (ZCHS) and for EVs (ZCEV) have been defined. These nodes contain essential attributes such as the state-of-charge of the EV, power limits and battery capacity. For controlling charging and discharging of the EV, the logical nodes for energy schedule, defined in [22], have been utilized. Since these extensions are outside the scope of this paper and subject for a later publication they will not be further described here.

C. *Case study conclusion*

Although μCHPs and EVs differ significantly in both function and composition, the cases presented in this section show how IEC 61850 with the REST interface can support communication with both types of DERs. The units can be monitored and controlled to optimize their behavior in relation to energy prices, user requirements and the state of the power system.

VI. Conclusion

The main aim of this paper has been to demonstrate how RESTful services, in conjunction with the data model of IEC 61850, can be used to increase interoperability and simplicity in DER communication. The IEC 61850 standard has a large and well-defined set of logical notes describing the various components and values of a DER unit. This paper has shown that the object reference path of the IEC 61850 data model can easily be mapped to the URL format in the resource-oriented approach used by REST. Besides offering an intuitive way of accessing information, REST also provides better interoperability and simplicity by shedding much of the complexity present in SOAP-based web services. The advantages of using IEC 61850 with REST have been demonstrated by building an IEC 61850 server and describing its functionality in two case studies.

There is no doubt that the ambition of actively integrating DERs into a smart grid constellation will rely on the utilization of contemporary web concepts and standards. This paper serves as an input to the identification of the ICTs capable of satisfying the communication requirements for the power system of the future.

References

[1] OPENADR - Outreach Collaborative homepage. http://www.openadrcollaborative.org/

[2] OPENAMD – standard sponsor. UCA International Users Group. http://www.ucaiug.org/

[3] EMIX – standard sponsor. OASIS Blue Initiative. http://www.oasis-blue.org/

[4] "Communication networks and systems in substations," *Commission Electrotechnique Internationale (IEC)*, vol. 1, July 2003.

[5] S. Heights, R. Mackiewicz, "Technical overview and benefits of the IEC 61850," Oct. 2006.

[6] "Manufacturing Message Specification (MMS)", Partl:Service Definition, Part2: Protocol Specification,International Standard IS0 9506, 1989.

[7] Edison project homepage. http://www.edison-net.dk/

[8] "IEC 61850-7-4, Basic communication structure - Compatible logical node classes and data object classes".

[9] E. B. Hauksson, A. B. Pedersen, "Enabling distributed energy resources in a virtual power plant using IEC-61850," Department of Informatics and Mathematical Modeling at the Technical University of Denmark, Master Thesis 2010.

[10] "IEC 61850-7-1, Basic communication structure for substation and feeder equipment Principles and models".

[11] "IEC 61850-7-2, Basic communication structure for substation and feeder equipment - Abstract communication service interface (ACSI),".

[12] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," PhD Thesis 2000. University of California, Irvine

[13] SOAP Specification. http://www.w3.org/TR/soap/

[14] IBM developerWorks, A. Rodriguez. RESTful Web services: The basics. https://www.ibm.com/developerworks/webservices/library/ws-restful/

[15] The application/json Media Type for JavaScript Object Notation. http://tools.ietf.org/html/rfc4627

[16] The Transport Layer Security (TLS) Protocol. http://tools.ietf.org/html/rfc5246

[17] JSON. http://www.json.org/

[18] WebHooks. http://www.webhooks.org/

[19] "IEC 61400-25-3, Communications for monitoring and control of wind power plants - Information exchange models".

[20] Internet X.509 Public Key Infrastructure: Certification Path Building. http://tools.ietf.org/html/rfc4158

[21] C. Binding, D. Gantenbein, B. Jansen, O. Sundström, P. Andersen, F. Marra, B. Poulsen, and C. Træholt, "Electric Vehicle Fleet Integration in the Danish EDISON Project - A Virtual Power Plant on the Island of Bornholm," IEEE Power & Energy Society, 2010 General Meeting , July 25-29, 2010, Minneapolis, Minnesota, USA.

[22] "IEC 61850-7-420, Basic communication structure - Distributed energy resources logical nodes".