About M2M standards

M2M and Open API

Manfred Sneps-Sneppe
Ventspils University College
Ventspils International Radioastronomy Centre
Ventspils, Latvia
manfreds.sneps@gmail.com

Dmitry Namiot
Lomonosov Moscow State University
Faculty of Computational Mathematics and Cybernetics
Moscow, Russia
dnamiot@gmail.com

***Abstract*—In this paper, we will discuss the current state of open APIs for M2M applications, as well as propose several possible changes and extensions. Our article based on open standards provided by ETSI. An open specification, presented as an Application Programming Interface (OpenAPI), provides applications with a rich framework of core network capabilities upon which to build services while encapsulating the underlying communication protocols. OpenAPI is a portable platform for services that may be replicated and ported between different execution environments and hardware platforms. We are proposing possible extensions for ETSI documents that let keep telecom development in sync with the modern approaches in the web development.**

Keywords-m2m; REST; open API; XML; web intents.

I. INTRODUCTION

As per classical definition from Numerex, Machine-to-Machine (M2M) refers to technologies that allow both wireless and wired systems to communicate with other devices of the same ability. M2M uses a device (such as a sensor or meter) to capture an event (such as temperature, inventory level, etc.), which is relayed through a network (wireless, wired or hybrid) to an application (software program), translates the captured event into meaningful information [1].

The next related acronym is Internet of things (IoT), referring to the networked interconnection of everyday objects [2]; it can be regarded as an extension of the existing interaction between humans and applications through the new dimension of "things" communication and integration. In IoT, devices are clustered together to create a stub M2M network, and are then connected to its infrastructure, i.e., the traditional "Internet of people" [3].

Considering M2M communications as a central point of Future Internet, European commission creates standardization mandate M/441 [4]. The Standardization mandate M/441, issued on 12th March 2009 by the European Commission to CEN, CENELEC and ETSI, in the field of measuring instruments for the development of an open architecture for utility meters involving communication protocols enabling interoperability, is a major development in shaping the future European standards for smart metering and Advanced Metering Infrastructures. The general objective of the mandate is to ensure European standards that will enable interoperability of utility meters (water, gas, electricity, heat), which can then improve the means by which customers' awareness of actual consumption can be raised in order to allow timely adaptation to their demands.

Besides the describing the current state of standards, our goal main here is the proposal for some new additions in M2M APIs architecture. We are going to propose web intents as add-on for the more traditional REST approach in order to simplify the development phases for M2M applications. The key moments in our proposals are: JSON versus XML, asynchronous communications and integrated calls.

The rest of the paper is organized as follows. Section II contains an analysis of M2M API standardization activities. In Section III, we consider Open API for M2M, submitted to ETSI. Sections IV and V are devoted to our offerings. In Section IV, we offer the never web tool – Web Intents for enhancement of M2M middleware. Sections V and VI are devoted to discussions.

II. THE CURRENT STATE OF M2M STANDARDS

Let us start from the basic moments. Right now, market players are offering own standards for M2M architecture. We refer to the recent ETSI TC M2M Workshop held on October 26-28, 2011. Figure 1 illustrates the basics of M2M infrastructure (as per ETSI) [5].

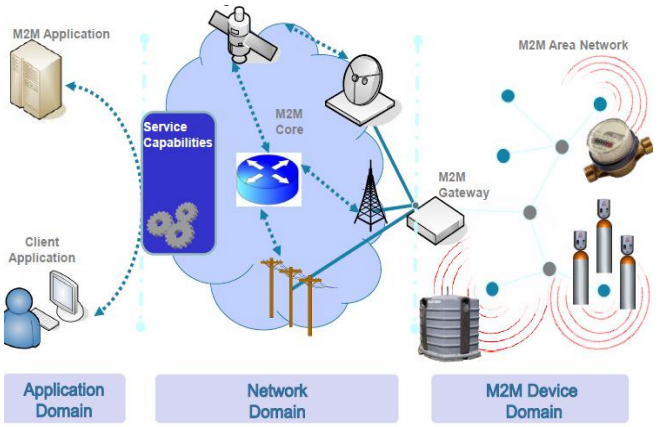

Figure 1. M2M infrastructure (as per ETSI)

 ISBN: 978-1-61208-193-9

The goals for M2M middleware are obvious. M2M middleware helps us with heterogeneity of M2M applications. Heterogeneity of service protocols inhibits the interoperation among smart objects using different service protocols and/or APIs. We assume that service protocols and API's are known in advance. This assumption prevents existing works from being applied to situations where a user wants to spontaneously configure her smart objects to interoperate with smart objects found nearby [6]. M2M API provides the abstraction layer necessary to implement interactions between devices uniformly. The M2M API provides the means for the device to expose its capabilities and the services it may offer, so that remote machines may utilize them. Consequently, such an API is necessary to enable proactive and transparent communication of devices, in order to invoke actions in M2M devices and receive the relating responses as well as the simplified management of resources.

ETSI is not the only source for the standardization in M2M area. Actually, ETSI has created a dedicated Technical Comittee for developing standards on M2M communications [7]. This structure aims at developing and maintaining an end-to-end architecture for M2M systems, as well as addressing various M2M communication considerations, such as naming, addressing, location, QoS, security, charging, management, application interfaces and hardware interfaces. Additionally, a major concern of the committee is to integrate sensor networks. The above-mentioned M/411 is just one example [12]. Other examples cover eHealth [8], Connected Consumer [9], City Automation [10] and Automotive Applications [11].

The 3rd Generation Partnership Project maintains and develops technical specifications and reports for mobile communication systems. Mobile networks are also concerned with the integration and support of M2M communications, as the nature of M2M systems is substantially differentiated than that of Human-to- Human services, i.e. plain telephone calls, which mobile networks originally addressed. Therefore, the 3GPP Technical Specifications Group dealing with Service and System Aspects [13], has issued a number of specifications dealing with requirements that M2M services and M2M communication imposes on the mobile network.

The Telecommunications Industry Association is the United States developing industry standards for a wide variety of telecommunication products. The standardization activities are assigned to separate Engineering Committees. The TR-50 Engineering Committee Smart Device Communications [14], has been assigned the task to develop and maintain physical-medium-agnostic interface standards, that will enable the monitoring and bi-directional communication of events and information between smart devices and other devices, applications or networks. It will develop a Smart Device Communications framework that can operate over different types of underlying transport networks (wireless, wired, etc.) and can be adapted to a given transport network by means of an adaptation/convergence layer.

The International Telecommunication Union as a specialized agency of the United Nations is responsible for IT and communication technologies. The Telecommunications Standardization Sector (ITU-T), covers the issue of M2M communication via the special Ubiquitous Sensor Networks-related groups [15]. ITU address the area of networked intelligent sensors.

Open Mobile Alliance (OMA) [16] develops mobile service enabler specifications. OMA drives service enabler architectures and open enabler interfaces that are independent of the underlying wireless networks and platforms. An OMA Enabler is a management object designated for a particular purpose. It is defined in a specification and is published by the Open Mobile Alliance as a set of requirements documents, architecture documents, technical specifications and test specifications. Examples of enablers would be: a download enabler, a browsing enabler, a messaging enabler, a location enabler, etc. Data service enablers from OMA should work across devices, service providers, operators, networks, and geographies.

As there are several OMA standards that map into the ETSI M2M framework, a link has been established between the two standardization bodies in order to provide associations between ETSI M2M Service Capabilities and OMA Supporting Enablers [17]. Specifically, the expertise of OMA in abstract, protocol-independent APIs creation, as well as the creation of APIs protocol bindings (i.e. REST, SOAP) and especially the expertise of OMA in RESTful APIs is expected to complement the standardization activities of ETSI in the field of M2M communications. Additionally, OMA has identified areas where further standardization will enhance support for generic M2M implementations, i.e. device management, network APIs addressing M2M service capabilities, location services for mobile M2M applications [18].

Actually, there should be a mapping of OMA service enablers to the ETSI M2M framework.

III. OPEN API FROM ETSI

This section describes an Open API for M2M, submitted to ETSI. By our opinion it is probably the most valuable achievement at this moment.

The Open API for M2M applications developed jointly in Eurescom study P1957 [19] and the EU FP7 SENSEI project makes. The OpenAPI has been submitted as a contribution to ETSI TC M2M [20] for standardization.

Actually, in this Open API we can see the big influence of Parlay specification. Parlay Group leads the standard, so called Parlay/OSA API, to open up the networks by defining, establishing, and supporting a common industry-standard APIs. Parlay Group also specifies the Parlay Web services API, also known as Parlay X API, which is much simpler than Parlay/OSA API to enable IT developers to use it without network expertise [21].

The goals are obvious, and they are probably the same as for any unified API. One of the main challenges in order to support easy development of M2M services and applications will be to make M2M network protocols "transparent" to

 ISBN: 978-1-61208-193-9

applications. Providing standard interfaces to service and application providers in a network independent way will allow service portability [22].

At the same time, an application could provide services via different M2M networks using different technologies as long as the same API is supported and used. This way an API shields applications from the underlying technologies, and reduces efforts involved in service development. Services may be replicated and ported between different execution environments and hardware platforms [23]

This approach also lets services and technology platforms to evolve independently. A standard open M2M API with network support will ensure service interoperability and allow ubiquitous end-to-end service provisioning.

The OpenAPI provide service capabilities that are to be shared by different applications. Service Capabilities may be M2M specific or generic, i.e., providing support to more than one M2M application.

Key points for Open API:

- It supports interoperability across heterogeneous transports
- ETSI describes high-level flow and does not dictate implementation technology
- It is message-based solution
- It combines P2P with client-server model
- It supports routing via intermediaries

At this moment, all points are probably well developed except the message-based decision. Nowadays, publish-subscribe method is definitely not among the favorites approaches in the web development, especially for heavy-loading projects.

Main API sections are:

- Subscription and Notification (e.g., Publish/Subscribe).
- Grouping.
- Transactions.
- Application Interaction: Read, Do, Observe.
- Compensation (micro-payment).
- Sessions.

Let us provide more details for Open API categories and make some remarks:

Grouping

A group here is defined as a common set of attributes (data elements) shared between member elements. On practice, it is about the definition of addressable and exchangeable data sets. Just note, as it is important for our future suggestions, there are no persistence mechanisms for groups

Transactions

Service capability features and their service primitives optionally include a transaction ID in order to allow relevant service capabilities to be part of a transaction. Just for the deploying transactions and presenting some sequences of operations as atomic.

In the terms of transactions management, Open API presents the classical 2-phase commit model. By the way, we should note here that this model practically does not work in the large-scale web applications. We think it is very important because without scalability we cannot think about “billions of connected devices”.

Application Interaction

The application interaction part is added in order to support development of simple M2M applications with only minor application specific data definitions: readings, observations and commands.

Application interactions build on the generic messaging and transaction functionality and offer capabilities considered sufficient for most simple application domains.

Messaging

The Message service capability feature offers message delivery with no message duplication. Messages may be unconfirmed, confirmed or transaction controlled. The message modes supported are single Object messaging, Object group messaging, and any object messaging; (it can also be Selective object messaging); thinking about this as Message Broker.

Event notification and presence

The notification service capability feature is more generic than handling only presence. It could give notifications on an object entering or leaving a specific group, reaching a certain location area, sensor readings outside a predefined band, an alarm, etc.

It is a generic form. So, for example, geo fencing [32] should fall into this category too.

The subscriber subscribes for events happening at the Target at a Registrar. The Registrar and the Target might be the same object. This configuration offers a publish/subscribe mechanism with no central point of failure.

Compensation

Fair and flexible compensation schemes between cooperating and competing parties are required to correlate resource consumption and cost, e.g., in order to avoid anomalous resource consumption and blocking of incentives for investments. The defined capability feature for micro-payment additionally allows charging for consumed network resources.

It is very similar to Parlay’s offering [33] for Charging API. Again, it is a big question from the modern large-scale applications point of view: shall we develop a special API for the compensations or create a rich logging functionality where the external log processing should be responsible for the things as charging.

 ISBN: 978-1-61208-193-9

Sessions

In the context of OpenAPI, a session shall be understood to represent the state of active communication between Connected Objects

OpenAPI is REST based, so, the endpoints should be presented as some URI's capable to accept (in this implementation) the basic commands GET, POST, PUT, DELETE.

Actually, ETSI uses the Smart Meter profile as 'proof of concept' for the M2M service platform in Release 1.

For example: requests execution of some function.

URI: http://{nodeId}/a/do
Method: POST

Request

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<appint-do-request
xmlns="http://eurescom.eu/p1957/openm2m">
<requestor>9378f697-773e-4c8b-8c89-
27d45ecc70c7</requestor>
<commands>
<command>command1</command>
<command>command2</command>
</commands>
<responders>9870f7b6-bc47-47df-b670-
2227ac5aaa2d</responders>
<transaction-
id>AEDF7D2C67BB4C7DB7615856868057C3</transactio
n-id>
</appint-do-request>
```

Response

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<appint-do-response
xmlns="http://eurescom.eu/p1957/openm2m">
<requestor>9378f697-773e-4c8b-8c89-
27d45ecc70c7</requestor>
<timestamp>2010-04-
30T14:12:34.796+02:00</timestamp>
<responders>9870f7b6-bc47-47df-b670-
2227ac5aaa2d</responders>
<result>200</result>
</appint-do-response>
```

Note that because we are talking about server-side solution, there is no problem with so called sandbox restrictions. But, it means of course, that such kind of request could not be provided right from the client side as many modern web applications do.

IV. THE MODERN WEB VS. OPEN API FROM ETSI

Let us describe the proposed standards from the modern web development points of view. As seems to us it is a correct approach, because Open API declares REST support right for the web development. In other words, support for web developers as the first class citizens is one of the obvious goals for ETSI.

It is almost impossible for developers to anticipate every new service and to integrate with every existing external service that their users prefer and thus they must choose to integrate with a few select APIs at great expense to the developer.

As per telecom experience we can mention here the various attempts for unified API that started, probably, with Parlay. Despite a lot of efforts, Parlay API's actually increase the time for development. It is, by our opinion, the main reason for the Parlay's failure.

Web Intents solves this problem. Web Intents is a framework for client-side service discovery and inter-application communication. Services register their intention to be able to handle an action on the user's behalf. Applications request to start an action of a certain verb (for example share, edit, view, pick etc.) and the system will find the appropriate services for the user to use based on the user's preference. It is the basic [24].

Intents play the very important role in Android Architecture. Three of the four basic OS component types - activities, services, and broadcast receivers - are activated by an asynchronous message called as intent.

Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your application or another.

Created intent defines a message to activate either a specific component or a specific type of component - an intent can be either explicit or implicit, respectively.

For activities and services, an Intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know). For example, our intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an Intent (for example, you can issue an intent to let the user pick a personal contact and have it returned to you - the return intent includes a URI pointing to the chosen contact) [25].

Going to M2M applications, it means that our potential devices will be able to present more integrated data for the measurement visualization for example. The final goal of any M2M based application is to get (collect) measurements and perform some calculations (make some decisions) on the collected dataset. We can go either via low level APIs or use (at least for the majority of use cases) some integrated solutions. The advantages are obvious. We can seriously decrease the time for development.

 ISBN: 978-1-61208-193-9

Web Intents puts the user in control of service integrations and makes the developers life simple.

Here, is the modified example for web intents integration for the hypothetical web intents example:

1. Register some intent upon loading our HTML document

```
document.addEventListener("DOMContentLoaded",
function() {
        var regBtn = document.getElementById("register");
        regBtn.addEventListener("click", function() {
     window.navigator.register("http://webintents.org/m2m",
undefined);    }, false);
```

2. Start intent's activity

```
        var startButton =
document.getElementById("startActivity");
        startButton.addEventListener("click", function() {
          var intent = new Intent();
          intent.action = "http://webintents.org/m2m";
            window.navigator.startActivity(intent);  }, false);
```

3. Get measurements (note – in JSON rather than XML) and display them in our application

```
        window.navigator.onActivity = function(data) {
          var output = document.getElementById("output");
          output.textContent = JSON.stringify(data);
        }; }, false);
```

Obviously, that it is much shorter than the long sequence of individual calls as per M2M Open API.

The key point here is *onActivity* callback, which returns JSON (not XML!) formatted data. In contrast, as per suggested M2M API, we should perform several individual requests, parse XML responses for the each of them and only after that make some visualization. Additionally, Web Intents based approach is asynchronous by its nature, so, we don need to organize asynchronous calls by our own.

Also, Web Intents approach let us bypass sandbox restrictions. In other words developers can raise requests right from the end-user devices, rather than always call the server. The server-side only solution becomes bottleneck very fast, and vice-versa, client side based request let developers deploy new services very quickly. Why do not use the powerful browsers in the modern smart-phones? At the end of the day Parlay spec were born in the time of WAP and weak phones. Why do we ignore HTML5 browsers and JavaScript support in the modern phones?

Generally speaking, we expect the initiatives from software companies that will opposite to telecom approach. For example, Paho project [26] (IBM et al.) directly declares the need to provide open source implementations of open and standard messaging protocols that support current and emerging requirements of M2M integration with Web and Enterprise middleware and applications. It will include client implementations for use on embedded platforms along with corresponding server support as determined by the community. This will enable a paradigm shift from legacy point-to-point protocols and the limitations of protocols like SOAP or HTTP into more loosely coupled yet determinable models. It will bridge the SOA, REST, Pub/Sub and other middleware architectures already well understood by Web 2.0 and Enterprise IT shops today, with the embedded and wireless device architectures inherent to M2M.

We think that XML days are over, JSON (and especially JSONP) is a key.

But, here goes the next big question: persistence.

V. DATA PERSISTENCE

The next question we would like to discuss relating to the M2M API's is probably more discussion able. Shall we add some persistence API (at least in the form of generic interface)?

The reasons are obvious – save the development time. Again, we should keep in mind that we are talking about the particular domain – M2M. In the most cases our business applications will deal with some metering data. As soon as we admit, that we are dealing with the measurements in the various forms we should make, as seems to us a natural conclusion – we need to save the data somewhere. It is very simple – we need to save data for the future processing.

So, the question is very easy – can we talk about M2M applications without talking about data persistence? Again, the key question is M2M. It is not abstract web API. We are talking about the well-defined domain.

As seems to us, even right now, before the putting some unified API in place, the term M2M almost always coexists with the term "cloud". And as we can see, almost always has been accompanied by the terms like automatic database logging, backup capabilities etc.

So, maybe this question is more for the discussions or it even could be provocative in the some forms, but it is: why there is no reference API for persistence layer in the unified M2M API? It is possible in general to create data gathering API without even mentioning data persistence? Shall we define cloud database API as a part of M2M standard or not?

The use of cloud computing means that data collection, processing, interface, and control can be separated and distributed to the most appropriate resource and device. In contrast, currently M2M implementations tend to combine data collection, processing, interface, and control.

Once transmitted to the cloud, data can be stored, retrieved and processed without having to address many of the underlying computing resources and processes traditionally associated with databases. For M2M applications, this type of virtualized data storage service is ideal [27]

As soon as ETSI standards define the interfaces, the developers we will be able to introduce various implementations. For example, it looks like NoSQL solutions are perfect fit for M2M applications.

These data stores operate by using key-value associations, which allows for a flatter non-relational form of association. NoSQL databases can work without fixed table schemes. It makes easy to store different data formats as well

 ISBN: 978-1-61208-193-9

as change and expand formats over time. It is very important for M2M applications (as well as for any type of applications tied with hardware). There are no "unified" devices in the real word. We simply cannot create an efficient schema that will serve all the devices (including new entrants). So M2M stores should be schema-less.

NoSQL databases could be easily scaled horizontally. Data is distributed across many servers and disks. Indexing is performed by keys route the queries to the datastore for the range that serves that key. This means different clusters respond to requests independently from other clusters, what increases throughput and response times. Quick adding new servers, database instances and disks and changing the ranges of keys can accommodate growth.

There are more then enough NoSQL systems on the market, they all have own APIs, so the question for M2M standardization body becomes even more important: shall we include the "unified" interface to data store into standard?

Suppose we do not as it is now. Does it mean that for OMA interfaces for example we will define own persistence approach each time we need data saving?

The topic that is tight linked with data persistence is a cloud. Obviously, for big data we should be able to integrate the information gathered via M2M into a large virtual information platform in a cloud [28]. This moment is completely missed in Open API. Shall we live with it, shall we pass problem to OMA enablers or what? As seems to us, this question should be addressed and answered.

We think, that in addition to developing open interfaces and standard system architectures, M2M ecosystems also need to establish a set of common software and hardware platforms to substantially reduce development costs and improve time to market.

VI. NEW SIGNALING DEMAND

Eventually, billions of devices — such as sensors, consumer electronic devices, smart phones, PDAs and computers — will generate billions of M2M transactions. For example, price information will be pushed to smart meters in a demand-response system. Push notifications will be sent to connected devices, letting a client application know about new information available in the network. The scale of these transactions will go beyond anything today's largest network operators have experienced. Signaling traffic will be the primary bottleneck as M2M communications increase. Alcatel-Lucent Bell Labs traffic modeling studies support this by comparing network capacity against projected traffic demand across multiple dimensions (such as signaling processing load on the radio network controller, air-interface access channel capacity, data volume and memory requirement for maintaining session contexts). The limiting factor is likely to be the number of session set-ups and tear-downs. For the specific traffic model and network deployment considered in the study, it is seen that up to 67 percent of computing resources in the radio network controller is consumed by M2M applications. [29].

How much of the traffic sent is network overhead? As an analysis carried on by A. Sorrevad [30] shows for ZigBee solution, a node is sending at least 40 Mbytes per year with the purpose of maintaining the network and polling for new data. The trigger data traffic for a year is much less - around1-10 Mbytes. Thus we see that the relationship between network and trigger traffic can range between 40:1 to 4:1 in a ZigBee solution that is following the home automation specification.

The traffic sent when maintaining a 6LoWPAN network is application specific. The relationship between network and trigger traffic can then be in the range 2:1 to 5:1.

As per [31], we can describe the several traffic-related issues for M2M. Time-controlled traffic is transmitted and received at periods of time that are defined well in advance. Time-tolerant traffic can support significant delays in data transmission and reception. This implies that the system can give lower access priority to or defer data transmission of time-tolerant traffic. When data traffic is "one way," it is only control signaling that is transmitted in the opposite direction. Digital signage and consumer devices are use cases where data may be device-terminated. One-way traffic may require changes to the network entry and addressing protocols. Extremely low latency requires that both network access latency and data transmission latency be reduced. This feature is required in many emergency situations (e.g., healthcare). Changes to the bandwidth request and network entry/re-entry protocols may be required to support extremely low latency. Infrequent traffic is common in many M2M use cases. This feature may enable sleep/idle mode improvements that save power and channel resources.

Due to the salient features of M2M traffic it may not be supported efficiently by present standards [31].

Why do we think also it is a time for traffic-related talks? By our opinion the reason is very simple. It is not obvious exactly how can we support transactional APIs (as per ETSI draft), without the dealing with the increased traffic. Simply – in our transactions we need the confirmation that device is alive, that operation has been performed, etc. All this is signaling traffic. Actually, this may lead to next provocative questions: do we really need transactional calls for all use cases? For example, the modern large-scale web applications (e.g. social networks) are not transactional internally.

VII. CONCLUSION

In this article, we briefly described the current state for the open unified M2M API from ETSI. We proposed some new additions – Web Intents as add-on for the more traditional REST approach. The main goal for our suggestions is the simplifying the development phases for M2M applications. The key advantages are JSON versus XML, asynchronous communications and integrated calls. Also we would like to point attention of readers to the couple of important questions that are not covered yet by our opinion: data persistence, clouds and signaling traffic.

VIII. ACKNOWLEDGEMENT

The paper is financed from ERDF's project SATTEH (No. 2010/0189/2DP/2.1.1.2.0/10/APIA/VIAA/019) being implemented in Engineering Research Institute «Ventspils

 ISBN: 978-1-61208-193-9

International Radio Astronomy Centre» of Ventspils University College (VIRAC).

REFERENCES

[1] Bob Emmerson, "M2M: The Internet of 50 Billion Devices", WinWin Magazine, Jan. 2010, pp.19-22.

[2] Commission of the European communities, Internet of Things in 2020, EPoSS, Brussels, 2008.

[3] Antoine de Saint-Exupery, Internet of Things – Strategic Research Roadmap, Sep. 15, 2009. http://www.internet-of-things-research.eu Retrieved: Jan, 2012

[4] Standartisation mandate to CEN, CENELEC and ETSI in the field of measuring instruments for the developing of an open architecture for utility meters involving communication protocols enabling interoperability, European Commission, M/441, 2009.

[5] ETSI Machine-to-Machine Communications http://www.etsi.org/website/technologies/m2m.aspx Revised: Feb, 2012

[6] Hyunho Park, Byoungoh Kim, Yangwoo Ko, and Dongman Lee, "InterX: A service interoperability gateway for heterogeneous smart objects" in in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference 21-25 March 2011 pp. 233 - 238.

[7] http://www.etsi.org/Website/Technologies/M2M.aspx Retrieved: Jan, 2012

[8] ETSI TR 102 732 V0.3.1, Machine to Machine Communications (M2M); Use cases of M2M applications for eHealth.

[9] ETSI TR 102 857 V0.3.0, Machine to Machine Communications (M2M); Use cases of M2M applications for Connected Consumer

[10] ETSI TR 102 897 V0.1.1, Machine to Machine Communications (M2M); Use cases of M2M applications for City Automation

[11] ETSI TR 102 898 V0.4.0, Machine to Machine Communications (M2M); Use cases of Automotive Applications in M2M capable networks.

[12] ETSI TR 102 691 V1.1.1, Machine-to-Machine communications (M2M); Smart Metering Use Cases

[13] 3GPP TS 22.368 V11.0.1, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for Machine-Type Communications (MTC); Stage 1, (Release 11)

[14] TR-50 standards http://www.tiaonline.org/standards/committees/committee.cfm?comm=tr-50 Retrieved: Jan, 2012

[15] Jea-Il Han, Anh-Duy Vu, Jin-Won Kim, Jun-Soo Jeon, Seung-Min Lee, and Young-Man Kim The fundamental functions and interfaces for the ITU-T USN middleware components Information and Communication Technology Convergence (ICTC), 2010 International Conference on 17-19 Nov. 2010 pp.: 226 – 231 Print ISBN: 978-1-4244-9806-2 DOI=10.1109/ICTC.2010.5674664

[16] OMA http://www.openmobilealliance.org/ Retrieved: Jan, 2012

[17] Niklas Blum, Irina Boldea, Thomas Magedanz, and Tiziana Margaria Service-oriented access to next generation networks: from service creation to execution Journal Mobile Networks and Applications archive Volume 15 Issue 3, June 2010 Kluwer Academic Publishers Hingham, MA, USA DOI=10.1007/s11036-010-0222-1 Retrieved Feb, 2012

[18] IoT project: http://www.iot-a.eu/public Retrieved Feb, 2012

[19] EURESCOM project P1957, Open API for M2M applications, http://www.eurescom.de/public/projects/P1900-series/P1957/. Retrieved Feb, 2012

[20] Draft ETSI TS 102 690 V0.13.3 (2011-07) Technical Specification

[21] Jong-choul Yim, Young-il Choi, and Byung-sun Lee Third Party Call Control in IMS using Parlay Web Service Gateway Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference Issue Date: 20-22 Feb. 2006 pp. 221 – 224

[22] Grønbæk I., Architecture for the Internet of Things (IoT): API and interconnect, The Second International Conference on Sensor Technologies and Applications, IEEE August 2008, DOI 10.1109/SENSORCOMM.2008.20, 809.

[23] Inge Grønbæk and Karl Ostendorf Open API for M2M applications In: ETSI M2M Workshop Oct. 2010

[24] Web Intents http://webintents.org/ Retrieved: Feb, 2012

[25] Android Developers http://developer.android.com/guide/topics/fundamentals.html Retrieved: Jan, 2012

[26] Paho project: http://eclipse.org/proposals/technology.paho/ Retrieved: Jan, 2012

[27] Cloud + Machine-to-Machine: http://www.readwriteweb.com/cloud/ 2011/03/cloud-machine-to-machine-disruptive-innovation-part-1p2.php Retrieved: Jan, 2012

[28] T. Osawa Practice of M2M Connecting Real World Things with Cloud Computing FUJITSU Sci. Tech. J. vol. 47 No. 4 pp. 401-407

[29] Harish Viswanathan, "Getting Ready for M2M Traffic Growth" http://www2.alcatel-lucent.com/blogs/techzine/2011/getting-ready-for-m2m-traffic-growth/ Retrieved: Jan, 2012

[30] A. Sorrevad M2M Traffic Characteristics, KTH Information and Communication Technology Master of Science Thesis Stockholm, Sweden 2009 TRITA-ICT-EX-2009:212 http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/091201-Anders_Orrevad-with-cover.pdf Retrieved: Feb, 2012

[31] Geng Wu, Shilpa Talwar, Kerstin Johnsson, Nageen Himayat, and Kevin D. Johnson, M2M: From Mobile to Embedded Internet IEEE Communications Magazine, April 2011 pp. 35-43

[32] A. Greenwald, G. Hampel, C. Phadke, and V. Poosala An economically viable solution to geofencing for mass-market applications Bell Labs Technical Journal Special Issue: Application Enablement Volume 16, Issue 2, September 2011 pp. 21–38,

[33] SunHwan Lim, JaeYong Lee, and ByungChul Kim Open API and System of Short Messaging, Payment, Account Management Based on RESTful Web Services Advanced Communication and Networking Communications in Computer and Information Science, 2011, Volume 199, pp. 66-75, DOI: 10.1007/978-3-642-23312-8_9

 ISBN: 978-1-61208-193-9