Open API for M2M Applications: What is Next?

Current state and development proposals

Manfred Schneps-Schneppe
Ventspils University College
Ventspils International Radio Astronomy 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*—This paper relates to telecommunication standards and describes the current status of open Application Programming Interface for M2M applications as well as proposes some changes and extensions. The European Telecommunications Standards Institute is going to provide open standards for the rapidly growing M2M market. An open specification, presented as an Open API, provides applications with a rich framework of core network capabilities upon which to build services while encapsulating the underlying communication protocols. Services may be replicated and ported between different execution environments and hardware platforms. We would like to discuss the possible extensions for ETSI proposals and describe the add-ons that, by our opinion, let keep telecom development inline with the modern approaches in the web development domain.**

Keywords-m2m; open api; rest; web intents.

I. INTRODUCTION

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].

Considering M2M communications as a central point of Future Internet, European commission creates standardization mandate M/441 [2]. 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.

Our goal is here to propose some new additions for M2M communications, namely, web intents, as add-on for the more traditional REST approach to simplify the development phases for M2M applications. The key advantages are JSON versus XML, asynchronous communications and integrated calls.

Right now, market players are offering own standards for M2M architecture [17].

Figure 1 illustrates the basics of M2M infrastructure (as per Cisco) [3].

The M2M infrastructure includes three primary domains: cloud, network, and edge devices. Each of these domains contains a specific anchor point which conducts the M2M signaling across the infrastructure. The M2M traffic has its own specific characteristics, such as low mobility and offline and online data transmission, which create new challenges for dimensioning the network. Service providers that are trying to customize their networks face the additional challenge of supporting traffic generated from residential and enterprise customer premises equipment (CPE).

Of course, there are several attempts to provide the standard set of software tools for M2M applications. These attempts are well explainable. Because M2M applications are directly linked to hardware devices than the portability of applications, the ability to bring new devices in system etc. become the key factors.

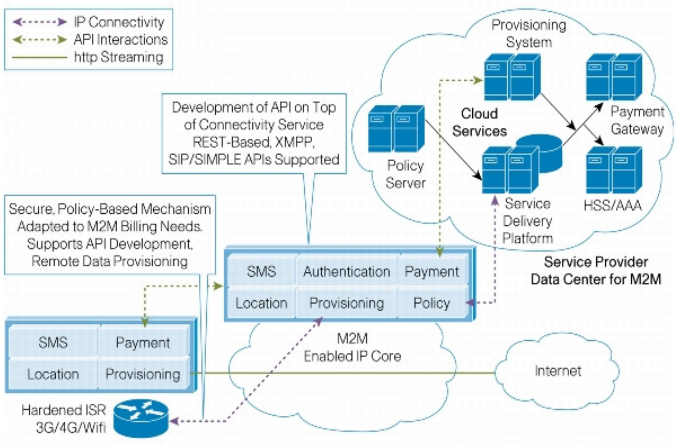

Figure 1. M2M infrastructure (as per Cisco)

Current customized M2M solutions and platforms tend to assume direct connectivity between the M2M core and devices, with no aggregators. However, linking residential and enterprise M2M gateways to an M2M-ready core opens new business models for service providers. M2M gateways can be bypassed when necessary.

In other words, what we can see now it is a growing interest to the M2M middleware.

 ISBN: 978-1-61208-199-1

Also, 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 API's. 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 [4].

Alcatel [5], for example, proposes the following conceptual view of M2M server (Fig. 2).

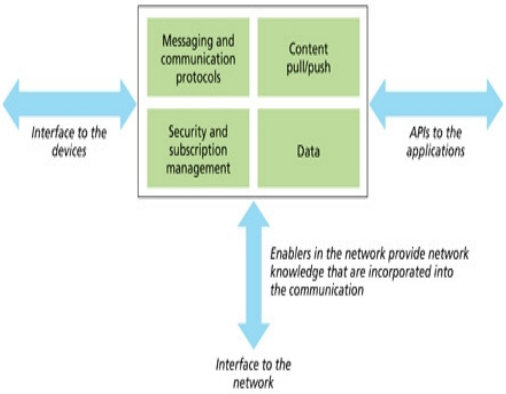

Figure 2. M2M server architecture

The gateway element should be located on the boundary between a wireless network and the Internet network used by application servers to communicate to a device. So, the M2M server can maintain sessions to application servers on one side, and to devices on the other side. In other words, it acts as a bridge, passing information from the application server to appropriate devices.

Web based architecture (or similar to web based) is about the common trend as we see. Many systems are offering for M2M developers the tools that developers are familiar with (e.g., from the previous projects, from the enterprise development, etc.), but the common denominators here are the standard protocols (REST, SOAP over HTTP) and nothing more. In other words, we can see the REST support in many (almost all) M2M frameworks, but the semantic for calls could be (almost always) different.

Of course, ETSI [2] is not the only source for the standardization in M2M area. The 3rd Generation Partnership Project maintains and develops technical specifications and reports for mobile communication systems [15]. 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 [16]. ITU address the area of networked intelligent sensors.

Also, we can see a growing interest to the cloud based M2M systems. For example, Axeda [14] offers cloud for M2M devices, including many traditional elements from the enterprise development world like business rules in orchestrations. [Fig. 3]

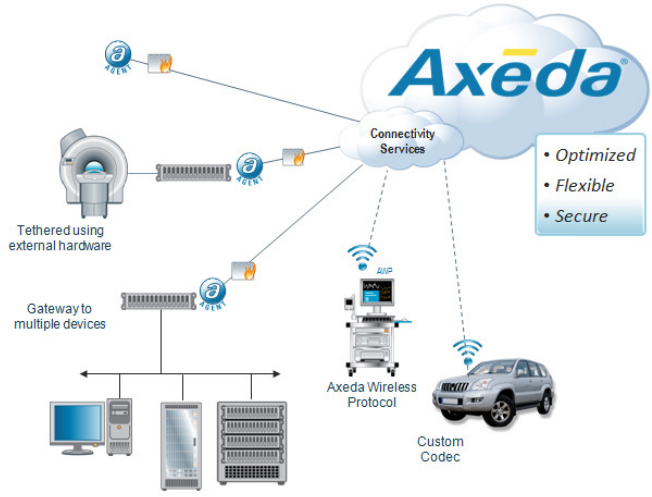

Figure 3. Axeda platform

Note that this system is actually very far from the European standards, despite the fact that it is also based on REST and SOAP as the ETSI standards. But in the same time AT&T has selected it as default M2M platform.

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

II. OPEN API FROM ETSI

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

The OpenAPI for M2M applications developed jointly in Eurescom study P1957 [6] and the EU FP7 SENSEI project [7] makes. The OpenAPI has been submitted as a contribution to ETSI TC M2M [8] 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 [9].

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 applications. Providing standard interfaces to service and application providers in a network independent way will allow service portability [10].

 ISBN: 978-1-61208-199-1

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 [11].

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 Open API provides 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

- and it supports routing via intermediaries

At this moment all point are probably not discussable 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.

Let us name the main Open API categories and make some remarks.

ETSI Open API categories	API contents	Comments
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). Think 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 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, by the way, to Parlay’s offering for Charging API.
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 (See an example below).

TABLE I. ETSI OPEN API CATEGORIES

 ISBN: 978-1-61208-199-1

A session 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</transaction-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.

III. WEB INTENTS VS. OPEN API FROM ETSI

Let us start from the basic. Users use many different services on the web to handle their day to day tasks, developers use different services for various tasks. In other words, our environment consists of connected applications. And of course, all they expect their applications to be connected and to work together seamlessly.

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 [9].

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 [12].

Going to M2M applications it means that our potential devices will be able to present more integrated 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 API's or use (at least for majority of use cases) some integrated solutions. The advantages are obvious. We can seriously decrease the time for development.

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);
```

 ISBN: 978-1-61208-199-1

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 that returns JSON (not XML!) formatted data. 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 lets 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?

IV. 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 on 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?

V. 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 [5].

How much of the traffic sent is network overhead? As an analysis carried on by A. Sorrevad [13] 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.

Why do we think it is a place for traffic talk? Because again, it is not clear completely how can we support transactional API's (as per ETSI draft [8]), 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.

VI. CONCLUSION

This article describes the current state for the open unified M2M API. Article proposes some new additions – web intents

 ISBN: 978-1-61208-199-1

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 to the couple of important questions that are not covered yet: data persistence and signaling traffic.

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

VIII. REFERENCES

[1] M. Chen, J. Wan, and F. Li Machine-to-machine communications: architectures, standards, and applications. KSII T Internet Inf 6(2): pp. 471–489, 2012

[2] 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.

[3] Managed and Cloud Services Insight Group Machine-to-Machine and Cloud Services: http://www.cisco.com/en/US/solutions/collateral/ns341/ns849/ns1098/white_paper_c11-663879.html. Retrieved: Mar, 2012

[4] H. Park, B. Kim, Y. Ko, and D. Lee "InterX: A service interoperability gateway for heterogeneous smart objects" in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference, 21-25 March 2011, pp. 233 – 238.

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

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

[7] Sensei project http://www.sensei-project.eu/. Retrieved: Feb, 2012

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

[9] J. Yim, Y. Choi, and B. 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.

[10] I. Grønbæk., "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.

[11] I. Grønbæk and K. Ostendorf "Open API for M2M applications" In: ETSI M2M Workshop Oct. 2010.

[12] Web Intents http://webintents.org/ Retrieved: Mar, 2012

[13] 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: Mar, 2012

[14] Axeda platform http://developer.axeda.com Retrieved: Feb, 2012

[15] 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)

[16] J. Han, A. Vu, J. Kim, J. Jeon, S. Lee, and Y. Kim; "The fundamental functions and interfaces for the ITU-T USN middleware components", Information and Communication Technology Convergence (ICTC), 2010 International Conference, 17-19 Nov. 2010, pp.: 226 – 231. Print ISBN: 978-1-4244-9806-2.

[17] K. Chang, A. Soong, M. Tseng, and X.Zhixian Global Wireless Machine-to-Machine Standardization. Internet Computing, IEEE, March-April 2011, Vol.: 15, Issue: 2, pp. 64 - 69

 ISBN: 978-1-61208-199-1