European Journal of Scientific Research
ISSN 1450-216X Vol. 94 No 3 January, 2013, pp.452-462

http://www.europeanjournalofscientificresearch.com

Web Intents as an Extension for M2M Open API

Dmitry Namiot
Lomonosov Moscow State University, Moscow, Russia
E-mail: dnamiot@gmail.com

Abstract

The paper briefly describes the current state of open APIs for M2M applications. As the main topic for review we consider an Application Programming Interface (Open API) from ETSI. It provides applications with a rich framework of core network capabilities upon which to build services while encapsulating the underlying communication protocols. Open API provides a portable platform for services that may be replicated and ported between different execution environments and hardware platforms. It is the more elaborated specification for M2M APIs to date. In this article we would like to propose the possible extension for ETSI documents, namely – Web Intents that, by our opinion, will keep telecom development in sync with the modern approaches in the web development.

Keywords: ETSI, m2m, REST, Open API, Parlay, XML, JSON, web intents

1. Introduction

Machine-to-machine communications is characterized by involving a large number of intelligent machines sharing information and making collaborative decisions without direct human intervention [1]. The typical M2M application 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).

The next close acronym is Internet of Things. The Internet of Things (IoT) is a concept in which the virtual world of information technology integrates seamlessly with the real world of things. The real world becomes more accessible through computers and networked devices in business as well as everyday scenarios [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. Actually, M2M API draft from ETSI is probably the most elaborated development in this area.

Besides the describing the current state of standards, our main goal 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 this proposal are: JSON versus XML for data exchange, asynchronous communications and integrated calls. During the last year, the research group author belongs to presented own vision for M2M standards (see Reference section). This paper summarizes our proposals for incorporating web development ideas and approaches into M2M world.

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 spec, submitted to ETSI. In Section IV we discuss a new proposal – Web Intents as an enhancement of M2M middleware.

2. The Current State of M2M Standards

M2M architecture from ETSI. Let us start from the basic moments. Right now market players are offering own standards for M2M architecture. We use partly our own article devoted to M2M standards review [5]. Discussions about some other aspects of M2M standards could be found in [6] and [7].

The high level architecture for M2M includes a Device and Gateway Domain and a Network domain. Actually, that schema is suggested by ETSI, but quite general and can be used for describing other frameworks too.

The Device and Gateway Domain is composed of the following elements: M2M Device, M2M Area Network and M2M Gateway.

M2M device is a device that runs M2M Application(s) using M2M Service Capabilities. It could be connected to Network Domain either directly or through M2M Gateway using the M2M Area Network. The M2M Device may provide service to other devices (e.g. legacy) connected to it that they are hidden from the Network Domain.

M2M Area Network provides connectivity between M2M Devices and M2M Gateways. Examples of M2M Area Networks include: M-Bus, Zigbee, Bluetooth, etc.

The M2M Gateway acts as a proxy for the Network Domain towards the M2M Devices that are connected to it. M2M Devices may be connected to the Networks Domain via multiple M2M Gateways. The M2M Gateway may provide service to other devices (e.g. legacy) connected to it that are hidden from the Network Domain. As an example an M2M Gateway may run an application that collects and treats various information (e.g. from sensors).

The Network Domain is composed of the following elements: Access Network, Core Network, M2M Service Capabilities and M2M applications.

Access Network: Network allows the M2M Device and Gateway Domain to communicate with the Core Network. Access Networks examples are xDSL, HFC, satellite, etc.

Core Network: provides:

- IP connectivity at a minimum and potentially other connectivity means.
- Service and network control functions.
- Interconnection (with other networks).
- Roaming.
- Different Core Networks offer different features sets.

Core Networks (CN) examples are 3GPP CNs, ETSI TISPAN CN and 3GPP2 CN.

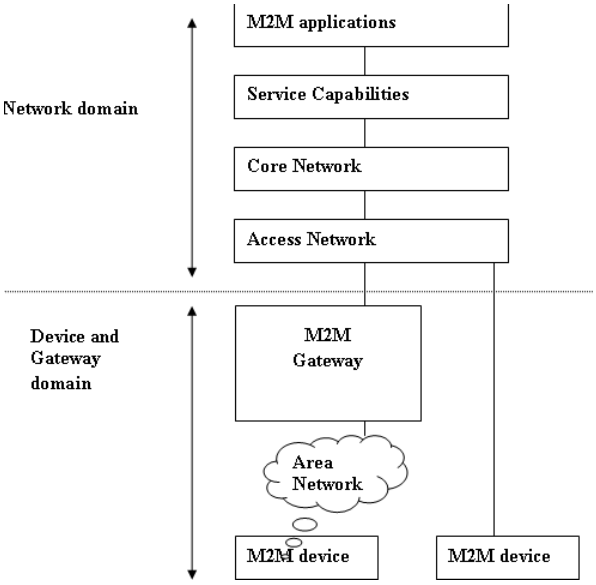
M2M Service Capabilities:

- Provide M2M functions that are to be shared by different Applications.
- Expose functions through a set of open interfaces.
- Use Core Network functionalities.
- Simplify and optimize application development and deployment through hiding of network specificities.

Actually the developer APIs are here. And our below suggested extensions target this level. M2M applications run the service logic and use M2M Service Capabilities accessible via an open interface.

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 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 own smart objects to interoperate with smart objects found nearby [9]. 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 [5]. Fugure 1 demonstrates the high level architecture for M2M.

Figure 1: M2M architecture from ETSI



On standartization mandate M/441. ETSI is not the only source for the standardization in M2M area. Actually ETSI has created a dedicated Technical Committee for developing standards on M2M communications. 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. ETSI insists that there is a specific for applied areas and it should be reflected in the appropriate standards.

We are using below parts from our paper [10] with the more detailed review of M2M standards.

ETSI TR 102 691 is probably the most elaborated document in ETSI's suite. It describes the following requirement areas to M2M applications:

Management - specifies requirements related to the management modes (malfunction detection, configuration, accounting, etc.).

Functional requirements for M2M services - describes functionalities-related requirements for M2M (data collection & reporting, remote control operations, etc.).

Security - covers the requirements for M2M device authentication, data integrity, privacy, etc.

Naming, numbering and addressing - provides the requirements relating to naming, numbering and addressing schemes specific to M2M.

As a something significant for this article we can highlight probably the list of potential new requirements to M2M systems (devices) listed here.

- A M2M device should be able to register it's capability information (e.g. access technology, its serial number, its accessible address, allowed user list, etc.) to the M2M System.
- M2M devices and M2M gateways should be able to perform access control that checks the access right of end-user.
- M2M devices should be able to communicate either directly or via M2M gateway.
- M2M device should be alternatively able to perform the access control of M2M devices.
- M2M devices and M2M gateways should be able to manage the scheduling of multiple accesses that multiple remote parties (i.e. end-users, M2M devices or M2M applications in M2M network, etc.) try to access one M2M device or one M2M gateway simultaneously.

Remark 1. Actually the first one remark is important: "register it's capability information". There is definitely a demand for some analogues of SNMP management, where capabilities could be defined in the abstract terms (like MIB – management information blocks).

On M2M Automotive Applications. Another group of requirements to M2M applications could be extracted from ETSI's documents devoted to Automotive Applications [12]. As per ETSI, M2M automotive applications encompass M2M use cases involving the automotive or transportation industries where the involved M2M communication modules may be embedded into a car or transportation equipment, for whatever purpose. This implies common requirements such as mobility management and environmental hardware constraints, despite the extended variety of applications addressed (insurance or road pricing, emergency assistance, fleet management, electric car charging management, traffic optimization, etc.). The new requirements listed here are:

- the capability of M2M Devices to receive, store, and execute scheduled measurements;
- the ability of Devices to poll and check for occurrence of events;
- the capability of Devices to autonomously establish a connection directly with a mobile telecommunication network;
- the capability for Devices to be able to maintain M2M communications while moving at high velocity and over a wide geographic area;
- the ability of devices to be able to be contacted ("called") directly by a mobile telecommunication network
- the inclusion of position-determination capability.

Remark 2. Note, that such a division, by our opinion, is one of the weakest points in the whole ETSI approach. From the developers point of view it is always good to have a small unified schema for all aspects. But ETSI's approach potentially leads to the huge set of different APIs. We saw the similar approach in Parlay [13] for example. It complicates the adoption for new development tools or even makes it impossible.

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 [14] 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 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. ITU-T covers the issue of M2M communication via the special Ubiquitous Sensor Networks-related groups. ITU address the area of networked intelligent sensors. As per ITU the system includes sensor nodes exchanging sensed data by wired or wireless processing (IP based sensor nodes with possibility of direct connection to NGN and non IP based nodes, often managed via gateway) and USN Gateway - a node which interconnects sensor networks with other networks

Open Mobile Alliance (OMA) 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.

Remark 3. 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. 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 [15].

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

3. 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.

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 [16].

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.

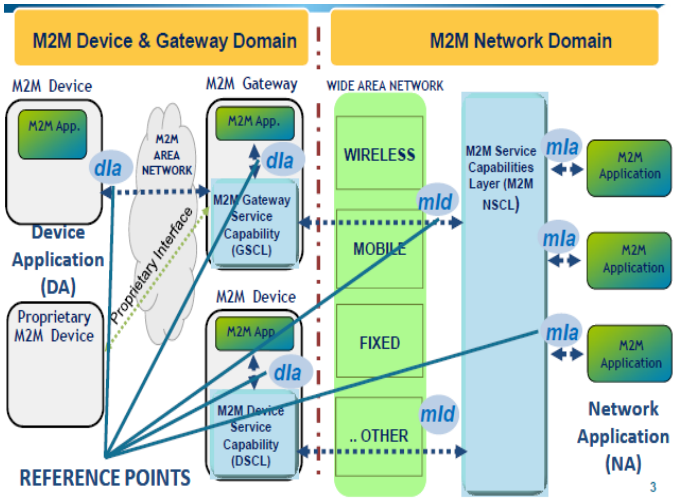
At the same time an application could provide services via different M2M networks using different technologies as long as the same API is supported and used. This way an API shields applications from the underlying technologies, and reduces efforts involved in service development. Services may be replicated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

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 relates to several interfaces of M2M architecture (Figure 2). For example:

- the interface between the platform and external service providers running their services remotely,
- the interface between the platform and the customer applying the features offered by the platform,
- a set of interfaces supporting additional functionality (installation support, access to remote databases, remote operation and management of platform), etc.

Figure 2: Interfaces from ETSI [8].



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

Remark 4. At this moment all points are probably not discussable except the message-based decision. Just introducing publish-subscribe approach without the efficient tools for data gathering may lead to the mass deployment of simple syndication tools based on repeated polling. One of the possible solutions here is adding peer to peer solutions. For example, it could be cooperative pooling. Several nodes could be assigned to periodically poll the same channel and shares updates detected by any polling node. So, many nodes polling with the same polling interval and randomly distributed polling times can detect updates much faster, as soon as our nodes can share updates with each other.

Main API sections for Services Capabilities Level are:

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

Remarks on Open API categories. 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 can not 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 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. 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.

Sessions. In the context of Open API a session shall be understood to represent the state of active communication between Connected Objects.

Open API 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.

Example 1: 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.

As per subscription based services, Open API provides ability to create callbacks. In other words, we can provide some unique URI in order to receive back asynchronous callbacks in the form of the HTTP calls.

4. Web Intents for Open API

Let us describe the proposed standards from the modern web development points of view. We think it is correct, 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. The history for this proposal is described in [18].

Lets us take two most important interfaces: Interactions and Messaging. They are assigned at the first hand for getting measurements. It is server side API. As per Open API vision the deployment includes the following steps:

- define the contact point (callback URL - x-etsi-contactURI header)
- perform the request
- proceed callbacks (HTTP requests) via callback URL

In other words it is a pure server side solution. We need to prepare CGI script for the each callback processing. Why do we ignore client side processing? Actually it is a main idea for our proposal – add client side processing for Open API. The main goal here is the simplifying of development. Time for the development is probably the key factor for any new API success.

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 solve this problem.

What is it? 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 [19].

Web Intents enable rich integration between web applications. Increasingly, services available on the web have a need to pass rich data back and forth as they do their jobs. Web Intents facilitate this interchange while maintaining the kind of loose coupling and open architecture that has proven so advantageous for the web. They reside purely client-side, mediated through the User Agent, allowing the user a great degree of control over the security and privacy of the exchanged data [20].

Any Intent is a user-initiated action delegated to be performed by a service. It consists of an "action" string which tells the service what kind of activity the user expects to be performed (e.g. "share" or "edit"), a "type" string which specifies the data payload the service should expect, and the data payload itself. So, we can replace callbacks (URLs) in Open API with JavaScript actions.

Web Intents is extensible by design. Neither the list of actions nor the list of media types are fixed. That is why intents can play an important role for semantic web too [21].

Intents play the very important role in Android Architecture. 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.

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.

We can re-phrase an original idea of Web Intents. M2M data logging application should be aware of a user's preferred editing Web application, rather than enforcing the specific one that the data logging happens to be integrated with.

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 Open API 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://location/resource/subs/id", undefined);
}, false);
```

2. Start intent’s activity (subscription in Open API)

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

3. Get measurements (notification in Open API) 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. 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, developers do not need to organize their own asynchronous schemes.

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. For example, right from mobile web browser. 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 dumb phones. Why do we ignore HTML5 browsers and JavaScript support in the modern phones?

Also, as it is showed above, this approach automatically introduces JSON versus XML communications. Again, JSON (and especially JSONP) is a preferred format for web development and should be welcomed by programmers.

5. Conclusion

In this article we briefly describe the current state for the open unified M2M API from ETSI. Our goal was to propose a new enhancement – web intents as add-on for the more traditional REST approach. The main goal for our suggestions is the simplifying of development phases for M2M applications. This proposal can substantially reduce development costs and accelerate the time to market. The key advantages are JSON versus XML usage for data transfer, asynchronous communications, integrated calls, client side deployment for M2M applications and the ability to bypass sandbox restrictions.

References

[1] R. Lu, X. Li, X. Liang, X. Shen, and X. Lin “GRS: The green, reliability, and security of emerging machine to machine communications”, Communications Magazine, IEEE, April 2011, Vol. 49 , No. 4, pp. 28-35

[2] D.Uckelmann, M.Harrison, and F. Michahelles “An Architectural Approach Towards the Future Internet of Things” ARCHITECTING THE INTERNET OF THINGS, 2011, pp. 1-24, DOI: 10.1007/978-3-642-19157-2_1

[3] A. de Saint-Exupery, “Internet of Things – Strategic Research Roadmap”, Sep.15, 2009. http://www.internet-of-things-research.eu

[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, EC, M/441, 2009.

[5] M. Sneps-Sneppe and D.Namiot “About M2M standards and their possible extensions” Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on, 25-27 April 2012 pp. 187-193 DOI: 10.1109/BCFIC.2012.6218001

[6] Sneps-Sneppe, M., & Namiot, D. (2012, April). About M2M standards. M2M and Open API. In ICDT 2012, The Seventh International Conference on Digital Telecommunications (pp. 111-117)

[7] Schneps-Schneppe, Manfred, and Dmitry Namiot. "Open API for M2M Applications: What is Next?." In AICT 2012, The Eighth Advanced International Conference on Telecommunications, pp. 18-23. 2012

[8] ETSI Machine-to-Machine Communications info and drafts http://docbox.etsi.org/M2M/Open/ Retrieved: Dec, 2012

[9] H. Park, B. Kim, Y. Ko, and D. 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.

[10] M. Sneps-Sneppe and D.Namiot “M2M Applications and Open API: What Could Be Next?” Internet of Things, Smart Spaces, and Next Generation Networking Lecture Notes in Computer Science Volume 7469, 2012, pp 429-439 DOI: 10.1007/978-3-642-32686-8_40
[11] http://www.etsi.org/Website/Technologies/M2M.aspx Retrieved: Dec, 2012
[12] ETSI TR 102 898 V0.4.0, M2M Communications; Use cases of Automotive Applications in M2M capable networks.
[13] A. Moerdijk and L.Klostermann “Opening the networks with Parlay/OSA: standards and aspects behind the APIs” Network, IEEE Vol. 17 , N 3 pp. 58 - 64
[14] 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)
[15] OMA http://www.openmobilealliance.org/ Retrieved: Mar, 2012
[16] 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, 20-22 Feb. 2006, pp. 221 – 224.
[17] I. Grønbæk and K. Ostendorf “Open API for M2M applications” In: ETSI M2M Workshop Oct. 2010.
[18] Daradkeh Yousef, Dmitry Namiot, and Manfred Sneps-Sneppe. "M2M Standards: Possible Extensions for Open API from ETSI." European Journal of Scientific Research 72.4 (2012): 628-637.
[19] P. Kinlan. Web Intents. Specification, Dec. 2010. Available at http://webintents.org/ Retrieved: Dec, 2012
[20] http://www.w3.org/TR/2012/WD-web-intents-20120626/ Retrieved: Dec, 2012
[21] R. Verborgh, T. Steiner, D. Deursen, S. Coppens, J. Vallés, and R.Walle Functional descriptions as the bridge between hypermedia APIs and the Semantic Web WS-REST '12 Proceedings of the Third International Workshop on RESTful Design, pp. 33-40