International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

FG M2M

D3.1 – Version 1.0
(04/2014)

ITU-T Focus Group on M2M Service Layer

M2M service layer: APIs and protocols overview

Focus Group Technical Report

SERIES OF FG M2M TECHNICAL REPORTS

Deliverable 0.1: M2M standardization activities and gap analysis: e-health

Deliverable 0.2: M2M enabled ecosystems: e-health

Deliverable 1.1: M2M use cases: e-health

Deliverable 2.1: M2M service layer: requirements and architectural framework

Deliverable 3.1: M2M service layer: APIs and protocols overview

Deliverable D3.1 “M2M service layer: APIs and protocols overview”

CONTENTS

M2M service layer: APIs and protocols overview

Scope

This Deliverable provides an overview of APIs and protocols for the M2M service layer and the related API and protocol requirements. At first, it describes the component based M2M reference model, including the reference points of the M2M service layer. Then, APIs and protocols for M2M are introduced, including existing APIs and protocols for M2M service layer and M2M protocol structure and stacks.

Finally, API and protocol requirements with respect to the M2M service layer are analysed.

1 Introduction

This decade is widely predicted to see the rise of machine-to-machine (M2M) communications over various networks. The M2M opportunity is diverse, dynamic, and growing fast. It provides connectivity for a huge variety of different devices and machines including utility meters, vehicles, sensors, and point of sale terminals, security devices, healthcare devices and many more. Every day, objects are becoming machines that can be addressed, recognized, localized, and controlled via communication networks and platforms.

An application programming interface (API) is a set of rules ('code') and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way a user interface facilitates interaction between humans and computers. In the M2M world, APIs provide the level of abstraction necessary to implement interactions uniformly. On the other hand, the data exchanged are inherently related to the protocol stack used in the communication.

A protocol is the special set of rules that end points in a communication network use when they communicate with each other. It is expected that M2M applications utilize standardized protocols in order to be widely deployable. Depending on the needs of the M2M application, different protocols may be utilized.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Deliverable. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Deliverable are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Deliverable does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Y.2060] ITU-T Recommendation Y.2060 (06/2012), *Overview of the Internet of things*

[ITU-T FG M2M D2.1] ITU-T FG M2M Deliverable D2.1 (2014), *M2M service layer: Requirements and architectural framework*

3 Definitions

3.1 Terms defined in this Deliverable

This Deliverable defines the following terms:

API: A particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another software program or set of resources.

4 Abbreviations and acronyms

This Deliverable uses the following abbreviations and acronyms:

API	Application Programming Interface
CoAP	Constrained Application Protocol
DA	Device Application
GA	Gateway Application
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
NA	Network Application
NAT	Network Address Translation
NAT-PMP	NAT Port Mapping Protocol
REST	Representational State Transfer
RTU	Remote Terminal Unit
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UPNP	Universal Plug and Play
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
XML	Extensible Markup Language
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks

5 Component based M2M reference model and its relationship with M2M service layer

5.1 Component based M2M reference model

The component based M2M reference model is shown in figure 1. It can be decomposed in five main components and one super-component:

− **Device:** It is the component that hosts the Device Applications [ITU-T FG M2M D2.1]. It can connect directly, or via the Gateway, to the Network. It may host M2M Service Layer (M2M SL) capabilities [ITU-T FG M2M D2.1]. When it does not contain the M2M SL capabilities, it is considered as a legacy device as described in [ITU-T FG M2M D2.1].

- **Gateway:** It is a component that may host M2M SL capabilities and Gateway Applications [ITU-T FG M2M D2.1] and acts as an intermediary between the Network and a legacy device as described in [ITU-T FG M2M D2.1].
- **Network:** It is a component, which does not host M2M SL capabilities, connecting Device, Gateway and Network Application Server with each other.
- **M2M Platform:** It is a component that hosts the M2M SL capabilities, and that can be used by one or more Application Servers. The M2M platform is part of the Network Application Server.

  NOTE - The following two clauses (5.2 and 5.3) provide some details about M2M platforms as they play a relevant role from a protocol/API point of view.
- **Application Server:** It is a component that hosts the Network Applications [ITU-T FG M2M D2.1]. The Application server is also part of the Network Application Server.
- **Network Application Server:** It is a super-component including both the Application Server and the M2M Platform.

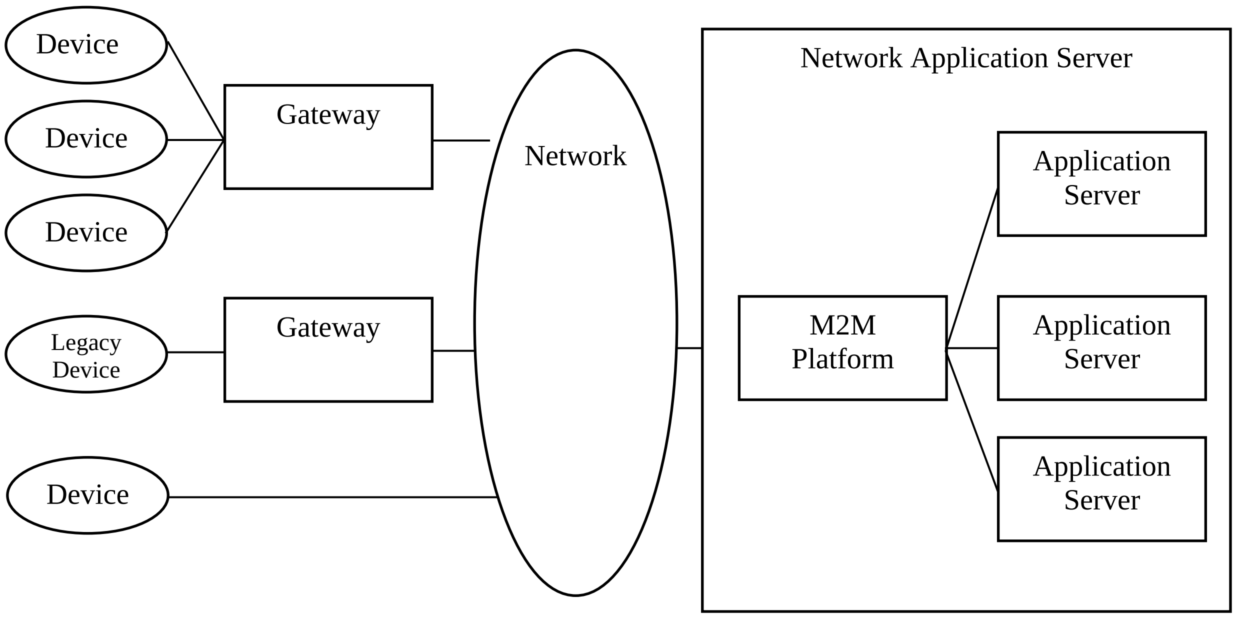

Figure 1: Component based M2M reference model

The components of the component based M2M reference model can communicate with each other using capabilities of multiple layers. NOTE - Recommendation ITU-T Y.2060 [ITU-T Y.2060] defines the IoT reference model as having four layers (Application layer, Service support and Application support layer, Network layer and Device layer).

5.2 M2M Platforms

Traditionally, M2M solutions have been conceived and deployed as ‘stovepipe’ (or standalone) solutions with the aim of improving (or enabling) a specific process, but without consideration of how these solutions might one day be integrated into a wider business context.

Today, M2M solutions are doing more than just monitoring the status of remote assets and equipment. They are gathering real-time data from millions of connected machines, for example, tractors, medical devices, vending machines, or storage tanks, and translating it into meaningful information for quick decisions, automated actions, and strategic analytics. The primary driver for M2M solutions is now enabling new services, rather than just improving operational efficiency/cost saving.

A platform is considered to be a group of technologies that are used as a base upon which applications, processes or other technologies are developed/delivered.

Creating a platform is usually a complex and delicate task, and needs to serve multiple purposes, but the primary purpose is to support and simplify the work of those who will be using/consuming this platform. M2M platforms have transformed the M2M market by making device data more accessible to application developers, and also by offering well-defined software interfaces and making APIs available so that application developers can readily integrate information sources and control parameters into their applications.

5.3 Types of M2M Platforms

Over the past decade, the M2M platform space has developed rapidly, and now includes the following broad platform functions:

- Connectivity Support: it encompasses all of the most fundamental tasks that must be undertaken to configure and support a machine-to-machine connection. In a mobile environment, such tasks include connection provisioning, usage monitoring, and some level of support for fault resolution.
- Service Enablement: it has extensive capabilities in terms of solution support, reporting and provision of a software environment and APIs to facilitate solution development. Together, Connectivity Support, and Service Enablement functions represent the ‘horizontal’ elements of the M2M platforms industry.
- Device Management: it has typically been aligned to single device manufacturers and potentially supports devices of multiple types and vendors connected through multiple networks. Device management platforms essentially exist to facilitate sales of devices (and device-centric solutions) where those devices typically require some form of non-standard systems support (reporting, management, etc.).
- Application Support: it is characterized by the provision of tailored solutions, encompassing connected devices potentially of multiple types, connected with multiple technologies, and connected to the networks of multiple CSPs (Communication Service Providers).
- Solution Provider: it should be regarded typically as an enabler for a large system development initiative, rather than as a standalone offering. These M2M platforms are generally used by systems integrators to support turnkey and client-specific solutions.

While many of these platform type implementations have some overlapping functionality (further complicating the M2M delivery ecosystem while simultaneously attempting to streamline it), the end goal is similar, i.e. mainly to make sure that the data collected from all of these machines and sensors are actually used to improve the business of the company investing in the collection.

5.4 Reference points of the M2M service layer in the component based M2M reference model

The purpose of this document is to provide guidelines about APIs and protocols handled by M2M service layer. It is necessary to make clear the reference points used by APIs and protocols. It is possible to clarify the reference points by referring to figure 4 of [ITU-T FG M2M D2.1], describing the M2M service layer reference points.

Figure 2 highlights the reference points that are in the scope of this document based on figure 4 of [ITU-T FG M2M D2.1]. Figure 2 focuses on reference points of the M2M service layer from a functional point of view.

There are three types of M2M applications on top of the M2M service layer: device application (DA), gateway application (GA) and network application (NA). DA, GA and NA reside, respectively, in device, gateway and network application server. All these applications use capabilities provided by the M2M service layer. These three types of applications are shown in the Application layer in figure 2.

Four reference points are identified for the ITU-T M2M service layer: D-SL, G-SL, A-SL and SL-SL.

- D-SL: reference point between DA and M2M service layer
- G-SL: reference point between GA and M2M service layer
- A-SL: reference point between NA and M2M service layer
- SL-SL: reference point between different M2M service layers

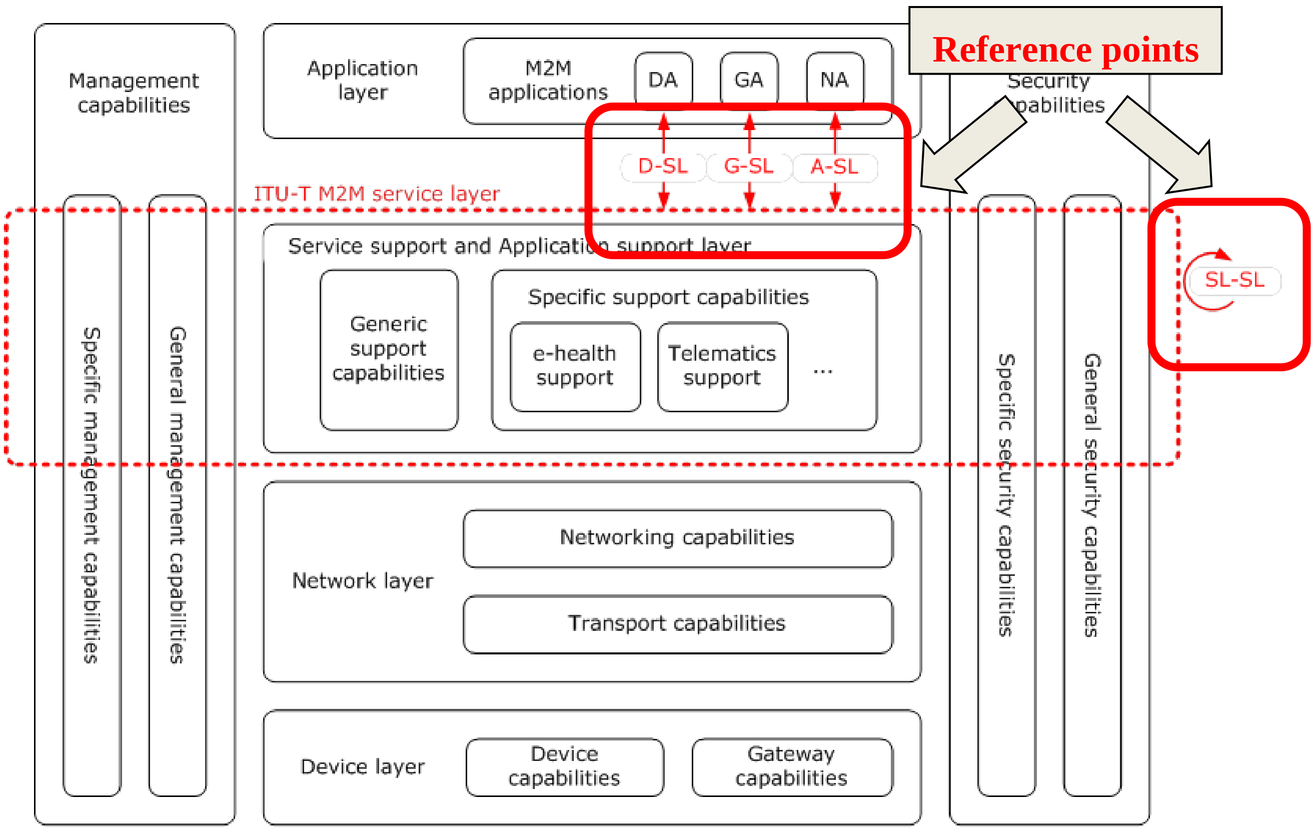

Figure 2: Reference points of the ITU-T M2M service layer

The four references points are shown in figure 3 with respect to the component based M2M reference model.

NOTE – Figure 3 shows the general case of devices providing also M2M SL capabilities (simply called “the SL function” in the following part of this document), a similar figure can be described for the case of legacy devices.

In line with what described in figure 5 of [ITU-T FG M2M D2.1], DA and the SL function are included in Device, GA and the SL function are included in Gateway, NA and the SL function are included in Network Application Server.

D-SL, G-SL, A-SL, SL-SL, as shown in figure 3, are established as reference points.

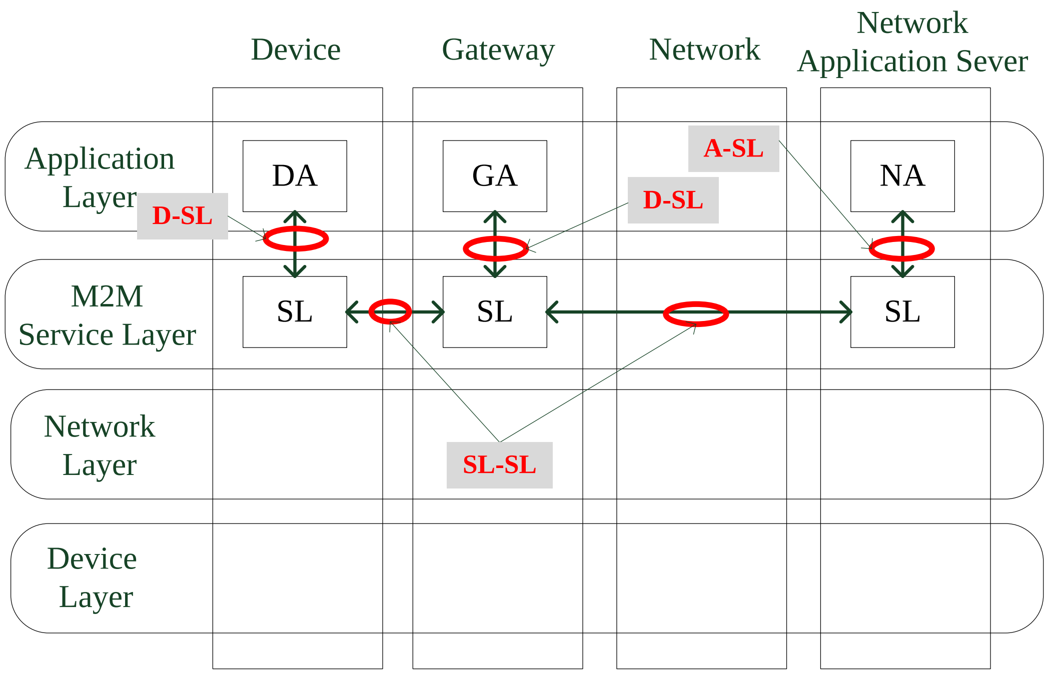

Figure 3: Reference points in the component based M2M reference model (case with no legacy device)

It is necessary to clarify requirements of each reference point which can be used by both Generic support capabilities and Specific support capabilities, which reside in the M2M service layer, in order to identify protocols and APIs to be used across these reference points.

6 APIs and protocols for M2M

6.1 API overview

API stands for "Application Programming Interface." An API is a set of commands, routines, functions, tools, and protocols which programmers can use when building software applications for a specific operating system. The APIs allow programmers to use predefined functions to interact with the operating system, instead of writing the functions from scratch. APIs can be implemented by an application to allow other applications to interact more easily and/or effectively with it.

An API shields applications from the underlying resources, and reduces efforts involved in service development. Services intended to be replicated and ported between different execution environments and hardware platforms. At the same time, services and technology platforms are allowed to evolve independently. A true value of a M2M enabled infrastructure comes not from collecting, integrating and analysing the information from all devices in order to achieve specific business purposes. A quick and effective way to accomplish this practical goal is to connect this M2M enabled infrastructure with the core systems and business processes of the infrastructure. These assets can be made accessible to the M2M enabled infrastructure via APIs. Standardized APIs aim to ensure service interoperability and allow ubiquitous end-to-end service provisioning.

Standardized APIs can provide efficiency in scale and, service production, and service development.

The M2M movement represents a significant shift for a number of industries. Connecting and digitizing data from disparate devices is usually disruptive to these industries. However, the foundational elements for the integration of M2M data and application services can be linked. Many companies have already begun to expose APIs that can be used in the context of M2M enabled applications. NOTE - These APIs may need to be tuned further in order to minimize the data payloads, adapt the data formats to fit the peculiarities of the connected devices, or include security policies that fit the profile of the data being exchanged. It makes sense then for the communication link between the M2M enabled infrastructure and the enterprise assets to be based on standardized APIs.

There are a number of distinct advantages to this approach:

- APIs allow for real-time integration of the M2M enabled infrastructure with the e-health information related storage area, removing costs associated with erroneous stored data and respecting regulatory boundaries of data storage.
- APIs provide a consistent approach for integrating the enterprise’s services, as well as those from the M2M enabled infrastructure providers, making skills and tools readily available
- APIs are Web-based, and they also enable the performance, scalability and security needed for high scale M2M deployments.

At a high level, APIs are ideal for the integration of a M2M enabled infrastructure, and indeed many readily available APIs can be used for this purpose. However, APIs have generally evolved in the context of real-time human interactions. There are some characteristics of M2M enabled interactions that differ, and must be considered when architecting M2M enabled applications:

- **Access control and security** – Much of the security and access control that is implemented for APIs assumes a human end user with specific permissions. A device-oriented security model is required to ensure appropriate control of the data flow. Different solutions may be also required depending on the access control requirements (e.g. API key model versus OAuth [IETF RFC6749]).
- **Synchronicity** – Many real-time APIs are synchronous. Many devices in a M2M enabled infrastructure require asynchronous communications for technical and business reasons. Existing APIs may need changes to handle these requirements.
- **Scale and bandwidth** – M2M enabled communications demand simultaneously high scalability to handle the proliferation of devices, while being constrained on bandwidth based on geographical deployment in locations atypical for IT. This requires flexibility in SLAs and optimization of APIs.

6.2 Design approach for M2M service layer APIs

There are mainly two design approaches for M2M service layer APIs.

6.2.1 Service-oriented architecture (SOA)

Service-oriented architecture (SOA) is a software design and software architecture design pattern based on discrete pieces of software providing application functionality as services to other applications.

A service is a self-contained unit of functionality. Services can be combined by other software applications to provide the complete functionality of a large software application. SOA makes it easy for computers connected over a network to cooperate. Every computer can run an arbitrary number of services, and each service is built in a way that ensures that the service can exchange information with any other service in the network without human interaction and without the need to make changes to the underlying program itself.

6.2.2 Resource-Oriented architecture (ROA)

Resource-oriented architecture (ROA) is a style of software architecture and programming paradigm for designing and developing software in the form of resources with "RESTful" interfaces. These resources are software components (discrete pieces of code and/or data structures) which can be reused for different purposes.

REST (Representational State Transfer) is an approach for getting information content from a Web site by reading a designated Web page that contains an XML (Extensible Markup Language) [XML 1.1] file that describes and includes the desired content. The REST approach is basically based on the web technologies such as transfer protocols like HTTP [IETF RFC2616], identification format as URI (Universal Resource Locator), representation formats such as XML or HTML (Hyper Text Markup Language) and content identifiers as MIME (Multipurpose Internet Mail Extensions) [IETF RFC2045] types. REST is neither a product nor a tool: it describes how a distributed software system can be architected. The design of a distributed system receives the stamp of approval of experts in REST, if the design meets a number of constraints: then the system design is called RESTful. REST is consistent with an information publishing approach that a number of Web log sites use to describe some aspects of their site content, called RSS (RDF Site Summary). RSS uses RDF (Resource Description Framework), a standard way to describe a Web site or other Internet resources.

6.3 Existing APIs and protocols for M2M service layer

A Protocol is a uniform set of rules that enable two devices to connect and transmit data to one another. Protocols determine how data are transmitted between computing devices, and over networks. Key capabilities of a protocol include the following: type of error checking to be used data compression method (if any), how the sending device will indicate that it has finished a message and how the receiving device will indicate that it has received the message.

The following are some of the existing APIs and protocols which can be considered for M2M service layer, but not limited to:

6.3.1 Modbus

Modbus is an application layer messaging protocol, positioned at level 7 of the OSI model, which provides client/server communication between devices connected on different types of buses or networks. Modbus is a serial communication protocol extensively used in SCADA (Supervisory Control And Data Acquisition) systems to establish a communication between RTU (Remote Terminal Unit) and devices.

Modbus Application protocol specification (04/2012): http://www.modbus.org [Modbus]

6.3.2 UPNP

UPnP technology defines architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. UPnP technology provides a distributed, open networking architecture that leverages TCP/IP and Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices.

UPnP Forum: http://www.upnp.org/

6.3.3 IGD

Internet Gateway Device (IGD) Standardized Device Control Protocol is an "edge" interconnect device between a residential Local Area Network (LAN) and the Wide Area Network (WAN), providing connectivity to the Internet. It is supported by some NAT routers. It is a common method of automatically configuring port forwarding.

UPnP Forum – Internet Gateway Device (IGD) (12/2010): http://www.upnp.org/ [IGD]

6.3.4 NAT-PMP

NAT Port Mapping Protocol (NAT-PMP) is a protocol for automating the process of creating Network Address Translation (NAT) port mappings. NAT-PMP allows a computer in a private network (behind a NAT router) to automatically configure the router to allow parties outside the private network to contact itself. NAT-PMP runs over UDP. It essentially automates the process of port forwarding. Included in the protocol is a method for retrieving the public IP address of a NAT gateway, thus allowing a client to make this public IP address and port number known to peers that may wish to communicate with it.

NAT Port Mapping Protocol (NAT-PMP) (04/2013): http://datatracker.ietf.org/doc/rfc6886/ [IETF RFC6886]

6.3.5 DPWS

Devices Profile for Web Services (DPWS) defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. DPWS is aligned with Web Services technology and includes numerous extension points allowing for seamless integration of device-provided services in enterprise-wide application scenarios. DPWS builds on the following core Web Services standards: WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, and further comprises WS-MetadataExchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing.

OASIS standard Devices Profile for Web Services (DPWS) (07/2009): http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html [DPWS]

6.3.6 BiTXML

The BiTXml communication protocol has been designed to implement a presentation level of the (OSI-based) communication stack reference, with the main goal to standardize the way commands and control information are exchanged for the specific target of M2Mcommunication demands (i.e. communication with generic devices with or without processing power on board - like sensors, actuators, as well as air conditioning systems, lifts, etc. or a combination of them).

BiTXml M2M communications Protocol (11/2007): http://www.bitxml.org/ [BiTXml]

6.3.7 CoAP

Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained networks and nodes for machine-to-machine applications such as smart energy and building automation. CoAP provides a method/response interaction model between application end-points, supports built-in resource discovery, and includes key web concepts such as URIs and content-types. CoAP easily translates to HTTP for integration with the Web while meeting specialized requirements such as multicast support, low overhead and simplicity for constrained environments.

Constrained Application Protocol (CoAP): http://datatracker.ietf.org/doc/draft-ietf-core-coap/ [IETF draft-ietf-core-coap]

6.3.8 SOAP

Simple Object Access Protocol (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.

W3C Recommendation - Simple Object Access Protocol (SOAP) (04/2007): http://www.w3.org/TR/soap12-part1/ [SOAP]

6.3.9 OAuth

The OAuth protocol's authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

The OAuth 2.0 Authorization Framework (10/2010): http://datatracker.ietf.org/doc/rfc6749/ [IETF RFC6749]

6.3.10 WebSocket

The WebSocket protocol enables two-way communication between clients running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).

The WebSocket Protocol (12/2011): http://datatracker.ietf.org/doc/rfc6455/ [IETF RFC6455]

6.4 M2M protocol structure and stacks

In order to introduce the M2M service layer related protocol layering and stacks, an example of protocol stacks in the component-based M2M reference model is shown in figure 4. The components of the component-based M2M reference model (device, gateway, M2M platform and application server) are all involved in the application related operations.

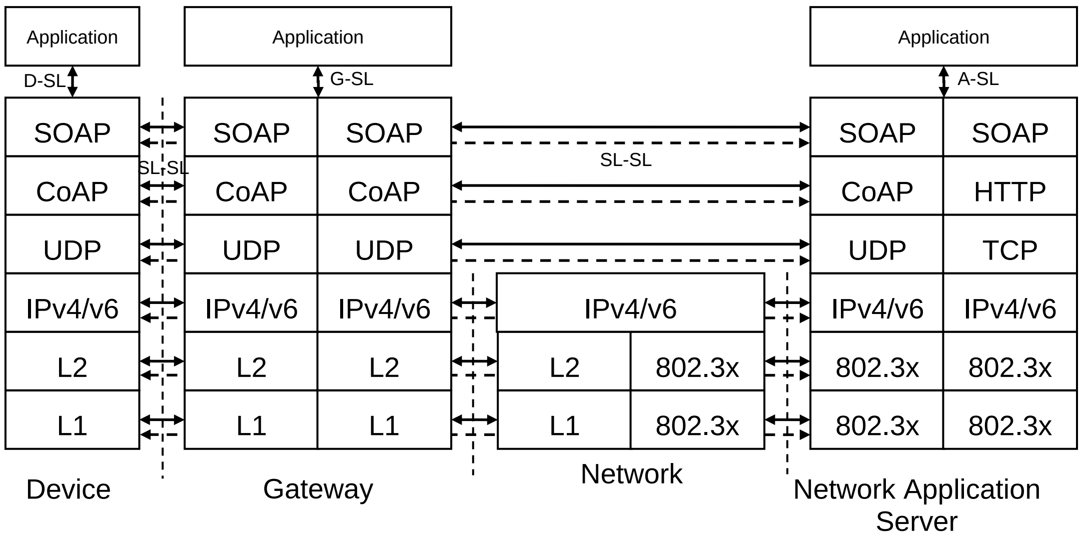

Figure 4: Example of protocol stacks in the component-based M2M reference model

There are 2 cases of connection types between devices and M2M platform. The first case is when a device communicates with M2M platform via a gateway(s), shown in figure 4 with continuous lines, and the second case is when a device communicates with M2M platform directly without a gateway(s), shown in figure 4 with dashed lines.

Figure 5 below shows a concrete application instance that utilizes the component-based M2M reference model for an e-health application. This concrete instance provides a specific example of M2M protocol stacks where, for example, a scale or a sphygmomanometer is used as device connected to a gateway, and a PHR server is used as application server. This example shows that IEEE11073/IEEE20601 [ISO/IEEE 11073-20601] is used for the SL-SL reference point between device and gateway, HL7v2/SOAP/HTTP [HL7v2] [SOAP] [IETF RFC2616] is used for the G-SL, A-SL and gateway-platform SL-SL reference points , and HL7v3/SOAP/HTTP [HL7v3] [SOAP] [IETF RFC 2616] is used for the A-SL reference point related to NA.

NOTE 1 - These protocol stacks are just one example; the appropriate protocols are required according to the devices and applications in use.

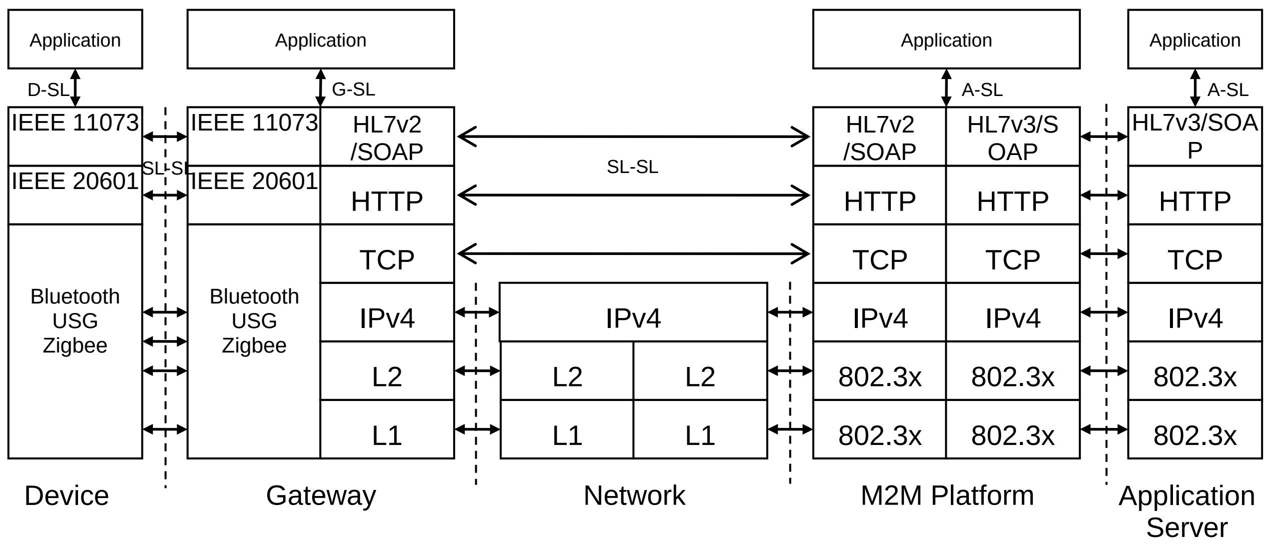

Figure 5: Example of M2M protocol stacks for e-health application (using gateway)

Another concrete application instance in figure 6 shows a specific example of M2M protocol stacks where, for example, a smart phone is used as device connected with the network without a gateway and a PHR server is used as application server. This example shows the usage of SOAP/CoAP [SOAP] [IETF draft-ietf-core-coap] for the D-SL and device-M2M platform SL-SL reference points, and SOAP/HTTP [SOAP] [IETF RFC2616] for the A-SL reference point related to NA.

NOTE 2 - These protocol stacks are just one example; the appropriate protocols are required according to the devices and applications in use.

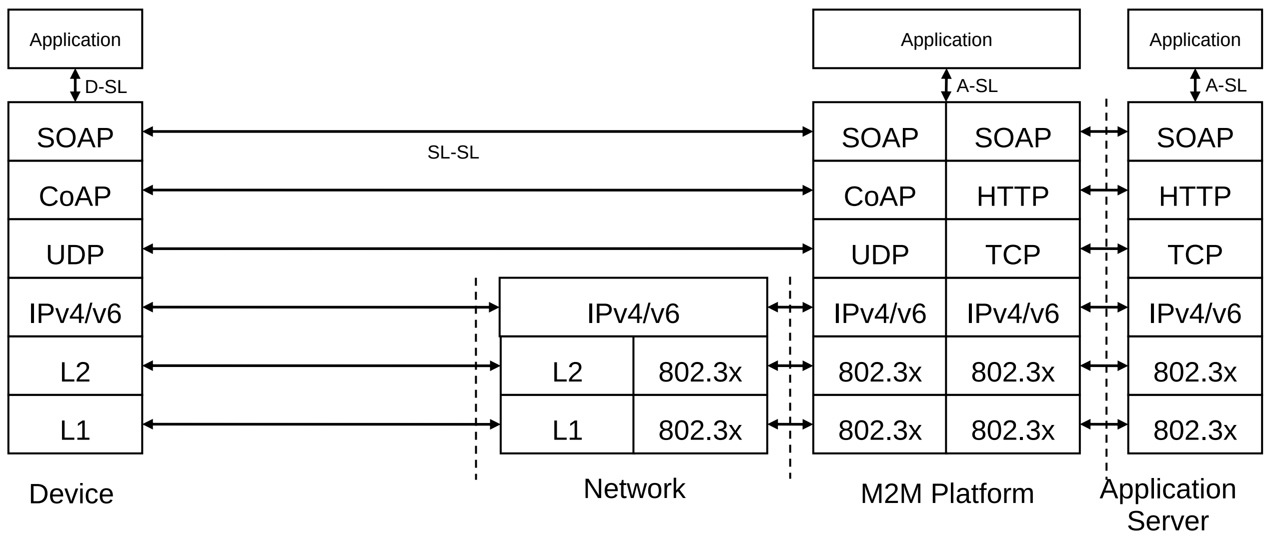

Figure 6: Example of M2M protocol stacks for e-health application (without gateway)

NOTE 3 – The present document considers specific cases for e-health application scenarios, but it will be necessary to consider other application scenarios (implying different types of devices, networks and security levels etc.) in the future.

7 Requirements of APIs and protocols with respect to the M2M service layer

7.1 Main requirements for support of e-health applications

Specific requirements for M2M service layer for support of e-health applications are described in [ITU-T FG M2M D2.1], clause 7.2 "e-health specific requirements". In the same deliverable, the capabilities which specifically support e-health applications in M2M service layer are described as “specific support capabilities”.

In the case of e-health applications, specific requirements should be considered for each reference point in addition to the common requirements related to generic support capabilities. These additional requirements for each reference point are based on the following e-health specific requirements as described in [ITU-T FG M2M D2.1]:

- Security for personal health information
- Privacy protection
- e-health device profile support
- Time synchronization and time stamping
- Audit trail support

7.2 Examples of attributes for APIs and protocols

Clause 6.3 lists existing APIs and protocols related to M2M service layer. Because of the large number of potential APIs and protocols, their classification and analysis is useful information for developers when selecting suitable protocols for M2M service layer. The following provides some examples of attributes to be considered for APIs and protocols with respect to the various interfaces:

- Interface for device – gateway, device – network application server and device – device: protocol load (information volume, connectionless/connection-oriented), routing capability, IP based/non IP based;
- M2M platform interface: communication security, scalability, real time capability, multitask capability, stateful/stateless;
- Application server interface: Internet compatibility (with other Internet services);
- Attributes common to all above indicated interfaces: expandability, usability, openness (including open source projects).

Bibliography

[BiTXml] BiTXml (11/2007), *M2M communications Protocol*, http://www.bitxml.org/

[DPWS] OASIS standard DPWS (07/2009), *Devices Profile for Web Services*, http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html

[HL7v2] HL7 Version 2, *HL 7 Standard version 2*, http://www.hl7.org/

[HL7v3] HL7 Version 3, *HL 7 Standard version 3*, http://www.hl7.org/

[IETF draft-ietf-core-coap] IETF draft-ietf-core-coap (06/2013), *Constrained Application Protocol (CoAP)*, http://datatracker.ietf.org/doc/draft-ietf-core-coap/

[IETF RFC2045] IETF RFC2045 (11/1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, http://datatracker.ietf.org/doc/rfc2045/

[IETF RFC2616] IETF RFC2616 (06/1999), *Hypertext Transfer Protocol -- HTTP/1.1*, http://datatracker.ietf.org/doc/rfc2616/

[IETF RFC2854] IETF RFC2854 (06/2000), *The 'text/html' Media Type*, http://datatracker.ietf.org/doc/rfc2854/

[IETF RFC6455] IETF RFC6455 (12/2011), *The WebSocket Protocol*, http://datatracker.ietf.org/doc/rfc6455/

[IETF RFC6749] IETF RFC6749 (10/2010), *The OAuth 2.0 Authorization Framework*, http://datatracker.ietf.org/doc/rfc6749/

[IETF RFC6886] IETF RFC6886 (04/2013), *NAT Port Mapping Protocol (NAT-PMP)*, http://datatracker.ietf.org/doc/rfc6886/

[IGD] UPnP Forum –IGD (12/2010), *Internet Gateway Device*, http://www.upnp.org/

[ISO/IEEE 11073-20601] Health informatics - Personal health device communication - Part 20601 (05/2010), *Application profile - Optimized exchange protocol*, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54331

[Modbus] Modbus (04/2012), *Application protocol specification*, http://www.modbus.org

[SOAP] W3C Recommendation SOAP (04/2007), *Simple Object Access Protocol*, http://www.w3.org/TR/soap12-part1/

[XML 1.1] W3C Recommendation XML 1.1 (08/2006), *Extensible Markup Language (XML) 1.1 (Second Edition)*, http://www.w3.org/TR/2006/REC-xml11-20060816/

Printed in Switzerland
Geneva, 2014