TTC技術レポート
Technical Report

TR-M2M-0009v0.7.0

oneM2M Protocol Analysis

2014 年 11 月 10 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TR-M2M-0009v0.7.0

oneM2M Protocol Analysis

＜参考＞ [Remarks]

１．国際勧告等の関連 [Relationship with international recommendations and standards]

本技術レポートは、oneM2M で作成された Technical Report 0009v0.7.0 に準拠している。

[This Technical Report is transposed based on the Technical Report 0009v0.7.0 developed by oneM2M.]

２．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

ONEM2M
TECHNICAL REPORT

Document Number	oneM2M TR-0009 v0.7.0
Document Name:	Technical Report - Protocol Analysis
Date:	12 June 2014
Abstract:	This document identifies protocols deployed in Industry segments, describes their use, then analyses protocols, their security aspects and their data models, with a view towards interoperability with oneM2M protocols.

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Contents

1 Scope

The present document will:

- Analyse the protocols, with consideration of security aspects (in cooperation with WG4 - Security) and data models (in cooperation with WG5 - Management, Abstraction & Semantics) widely considered for use within oneM2M's target industry segments
- Create a list of those protocols with which oneM2M could encapsulate and/or interoperate

Noting that: Widely used protocol mappings may be candidates for oneM2M work;
Industry or application-specific protocol mappings to oneM2M may be done by external organizations

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

2.1 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] oneM2M Drafting Rules, http://member.onem2m.org/Static_pages/Others/Rules_Pages/oneM2M-Drafting-Rules-V1_0.doc

[i.2] oneM2M TS-0002 oneM2M Requirements

[i.3] IBM MQ Telemetry Transport (MQTT) v3.1 Protocol Specification

[i.4] IETF draft-ietf-core-coap-18 Constrained Application Protocol

[i.5] TIA-4940-020, Smart Device Communications Protocol Aspects TIA TR-50

[i.6] IETF RFC6120 XMPP: Core

[i.7] IETF RFC2616 Hypertext Transfer Protocol -- HTTP/1.1

[i.8] IETF RFC6690 Constrained RESTful Environments (CoRE) Link Format

[i.9] IETF RFC4944 Transmission of IPv6 Packets over IEEE 802.15.4 Networks

[i.10] IETF RFC0768 User Datagram Protocol

[i.11] IETF RFC6347 Datagram Transport Layer Security Version 1.2

[i.12] W3C Extensible Markup Language (XML) 1.0 (Fifth Edition)

[i.13] IETF RFC4627 The application/json Media Type for JavaScript Object Notation (JSON)

[i.14] IETF RFC6121 Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, March 2011.

[i.15] IETF RFC6122 Extensible Messaging and Presence Protocol (XMPP): Address Format, March 2011

[i.16] IETF RFC4492 Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS), May 2006

[i.17] IETF RFC4422 Simple Authentication and Security Layer (SASL).

[i.18] XMPP Standards Foundation XEP-0016 Privacy Lists

[i.19] XMPP Standards Foundation XEP-0030 Service Discovery

[i.20] XMPP Standards Foundation XEP-0045 Multi-user conferencing service

[i.21] XMPP Standards Foundation XEP-0060 Publish-Subscribe service

[i.22] XMPP Standards Foundation XEP-0079 Advanced-Message Processing

[i.23] XMPP Standards Foundation XEP-0080 User Location

[i.24] XMPP Standards Foundation XEP-0136 Message Archiving

[i.25] XMPP Standards Foundation XEP-0138 Stream Compression

[i.26] XMPP Standards Foundation XEP-0149 Time Periods

[i.27] XMPP Standards Foundation XEP-0166 Jingle

[i.28] XMPP Standards Foundation XEP-0167 Jingle RTP Sessions

[i.29] XMPP Standards Foundation XEP-0177 Jingle Raw UDP Transport Method

[i.30] XMPP Standards Foundation XEP-0198 Stream Management

[i.31] XMPP Standards Foundation XEP-0199 XMPP Ping

[i.32] XMPP Standards Foundation XEP-0124 Bidirectional-streams Over Synchronous HTTP (BOSH)

[i.33] XMPP Standards Foundation XEP-0206 XMPP Over BOSH

[i.34] XMPP Standards Foundation XEP-0203 Delayed Delivery

[i.35] XMPP Standards Foundation XEP-0322 Efficient XML Interchange (EXI) Format for XMPP

[i.36] XMPP Standards Foundation XEP-0323 Internet of Things – Sensor Data

[i.37] XMPP Standards Foundation XEP-0324 Internet of Things – Provisioning

[i.38] XMPP Standards Foundation XEP-0325 Internet of Things – Control

[i.39] XMPP Standards Foundation XEP-0326 Internet of Things – Concentrators

[i.40] IETF RFC6455 The WebSocket Protocol, 2011

[i.41] OMG Data Distribution Service for Real-time Systems Version 1.2, January 2007

[i.42] OMG The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification, Version 2.1, June 2008

[i.43] - IEEE Standards for Electric Power Systems Communications – Distributed Network Protocol (DNP3), IEEE Std 1815 – 2012

[i.44] XMPP Standards Foundation XEP-0127 Common Alerting Protocol (CAP) over XMPP

[i.45] XMPP Standards Foundation XEP-0115 Entity Capabilities

[i.46] XMPP Standards Foundation XEP-0248 PubSub Collection Nodes

[i.47] XMPP Standards Foundation XEP-0072 SOAP over XMPP

[i.48] UPnP Forum, www.upnp.org .

[i.49] International Organization for Standardization (ISO). Located at www.iso.org

[i.50] International Electrotechnical Commission (IEC). Located at www.webstore.iec.ch

[i.51] ISO/IEC UPnP press release. Available at: http://www.iso.org/iso/news.htm?refid=Ref1500

[i.52] UPnP Device Architecture documents.
Available at http://upnp.org/sdcps-and-certification/standards/device-architecture-documents

[i.53] http://www.perfdynamics.com/Manifesto/USLscalability.html

[i.54] http://www.rti.com/products/dds/benchmarks-cpp-linux-scalability.html#THRUSCAL

[i.55] http://www.slideshare.net/Angelo.Corsaro/dscriptjs

[i.56] OMA RESTful Network API for FileTransfer
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_FileTransfer-V1_0-20130612-C.zip

[i.57] OMA RESTful Network API for Presence
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Presence-V1_0-20130212-C.zip

[i.58] OMA RESTful Network API for Notification Channel
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Presence-V1_0-20130212-C.zip

[i.59] OMA RESTful Network API for Chat
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Chat-V1_0-20131203-D.zip

[i.60] OMA RESTful Network API for Short Messaging
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Chat-V1_0-20131203-D.zip

[i.61] OMA RESTful Network API for Third Party Call
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_ThirdPartyCall-V1_0-20130212-C.zip

[i.62] OMA RESTful Network API for Address Book
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_ThirdPartyCall-V1_0-20130212-C.zip

[i.63] OMA RESTful Network API for Messaging
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Messaging-V1_0-20130709-C.zip

[i.64] OMA RESTful Network API for Payment
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_Payment-V1_0-20130924-A.zip

[i.65] OMA RESTful Network API for Device Capabilities
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_DeviceCapabilities-V1_0-20130924-A.zip

[i.66] OMA RESTful Network API for Audio Call
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_AudioCall-V1_0-20130212-C.zip

[i.67] OMA RESTful Network API for Call Notification
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_CallNotification-V1_0-20130212-C.zip

[i.68] OMA RESTful Network API for Terminal Status
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_TerminalStatus-V1_0-20131008-C.zip

[i.69] OMA RESTful Network API for Image Share
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_ImageShare-V1_0-20130605-C.zip

[i.70] OMA RESTful Network API for Terminal Location
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_TerminalLocation-V1_0-20130924-A.zip

[i.71] OMA RESTful Network API for Video Share
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_VideoShare-V1_0-20130517-C.zip

[i.72] OMA RESTful Network API for Video Share
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_CustomerProfile-V1_0-20130305-C.zip

[i.73] OMA RESTful Network API for ACR (Anonymous Customer Reference)
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_ACR-V1_0-20130625-C.zip

[i.74] OMA RESTful Network API for Capability Discovery
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_CapabilityDiscovery-V1_0-20130701-C.zip

[i.75] OMA RESTful Network API for Converged Address Book
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-RD-REST_NetAPI_CAB-V1_0-20130702-A.zip

[i.76] OMA RESTful Network API for Push
http://member.openmobilealliance.org/ftp/Public_documents/CD/Permanent_documents/OMA-TS-REST_NetAPI_Push-V1_0-20131029-A.zip

[i.77] OMA RESTful Network for Network Message Storage (Draft)
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/REST_NMS/Permanent_documents/OMA-TS-REST_NetAPI_NMS-V1_0-20131209-D.zip

[i.78] OMA RESTful Network for Network Message Storage (Draft)
http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-TS-REST_NetAPI_VVoIP-V1_0-20131120-D.zip

[i.79] Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives, Future Internet 2010, 2, 96-125, Open Access Journal,
http://www.mdpi.com/journal/futureinternet

[i.80] ISA100 Wireless Compliance Institute (WCI), http://www.isa100wci.org

[i.81] ISA100.11a, Wireless Systems for Industrial Automation: Process Control and Related Applications, September 2009. Also, IEC 62734

[i.82] WCI Members: http://www.isa100wci.org/en-US/About-WCI/Member-Roster

[i.83] ISA100 Wireless Product Listing: http://www.isa100wci.org/en-US/End-User-Resources/ISA100-Wireless-Registered-Products

[i.84] ISA100 Success Stories: http://www.isa100wci.org/en-US/End-User-Resources/Company-Success-Stories

[i.85] Wireless standards in action, A Closer look at ISA-100.11a,
http://www.isa.org/InTechTemplate.cfm?template=/ContentManagement/ContentDisplay.cfm&ContentID=84319

[i.86] HART Communication Foundation, www.hartcomm.org

[i.87] HART Protocol Specifications: http://www.hartcomm.org/hcf/documents/documents_spec_list.html

[i.88] Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives, Future Internet 2010, 2, 96-125, Open Access Journal,
http://www.mdpi.com/journal/futureinternet

[i.89] Evolution of wireless sensor networks for industrial control, Arthur Low, Technology Innovation Management Review, May 2013, http://timreview.ca/article/682 [i.90] Industrial communication networks - Wireless communication network and communication profiles - WirelessHART® IEC62591

[i.91] Bluetooth 4.1 technical specification: https://www.bluetooth.org/en-us/specification/adopted-specifications

[i.92] OASIS MQTT Version 3.1.1 Committee Specification Draft 01 18 May 2014

[i.93] W3C XML Schema Definition Language (XSD) 1.0, May 2001

[i.94] W3C XML Schema Definition Language (XSD) 1.1, 5 April 2012

[i.95] W3C Web Application Description Language (WADL) 31 August 2009

3 Abbreviations and acronyms

3.1 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CHG	Charging Requirement
OPR	Operational Requirement
PHY	Physical layer of the OSI model

3.2 Acronyms

For the purposes of the present document, the following acronyms apply:

6LOWPAN	IPv6 over Low power Wireless Personal Area Networks
ACT	Availability for Concurrent Transactions
ADSU	Application Service Data Unit
AMI	Advanced Metering Infrastructure
API	Application Programming Interface
ASCI	American Standards Compliance Institute
AV	Audio Video (in UPnP context)
BLE	Bluetooth Low Energy
BOSH	Bidirectional-streams Over Synchronous HTTP
CIP	Common Industrial Protocol
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments
CP	Control Point
CRPR	Communications Request Processing Requirement
CRUD	Create Read Update Delete
DCP	Device Control Protocol
DCPS	Data Centric Publish Subscribe
DDD	Device Description Document
DDS	Data Distribution Service
DI	Distributed Intelligence
DNP	Distributed Network Protocol
DNP3	Distributed Network Protocol - 3rd Generation
DTLS	Datagram Transport Layer Security
D2D	Device to device
EDR	Enhanced Data Rate
EXI	Efficient XML Interchange
GAP	Generic Access Profile
GATT	Generic ATTribute Profile
GENA	General Event Notification Architecture
GUI	Graphical User Interface
HART	Highway Addressable Remote Transducer Protocol
HATEOAS	Hypermedia As The Engine Of Application State
HCF	HART Communication Foundation

HIDS	Human Interface Devices
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Devices
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IBM	International Business Machines
IGD	Internet Gateway Device
IoT	Internet of Things
IP	Internet Protocol
IPSO	Internet Protocol for Smart Objects
IPR	Intellectual Property Rights
ISA	Industrial Society of Automation
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MAC	Media Access Control
MIME	Multipurpose Internet Mail Extensions
MQTT	Message Queuing Telemetry Transport
M2M	Machine to Machine
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OSR	Overall System Requirement
P2M	Person to Machine
P2P	Peer to Peer
PIM	Platform Independent Model
PLC	Programmable Logic Controller
PC	Personal Computer
QoS	Quality Of Service
RAM	Random Access Memory
REST	REpresentational State Transfer
RF	Radio Frequency
RFC	Request For Comments
ROM	Read Only Memory
RPC	Remote Procedure Call
RTPS	Real Time Publish Subscribe
RTU	Remote Terminal Unit
SASL	Simple Authentication and Security Layer
SCADA	Supervisory Control And Data Acquisition
SDO	Standards Development Organisation
SER	Security Requirement
SIG	Special Interest Group
SIP	Session Initiation Protocol
SMR	Semantics Requirement
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SPI	Service Plugin Interface
SSDP	Simple Service Discovery Protocol
SSL	Secure Sockets Layer
TAG	Technical Architecture Group
TIA	Telecommunications Industry Association
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UAV	Unmanned Aerial Vehicle
UCA	UPnP Cloud Architecture
UCD	Unicast Connectionless Data
UDA	UPnP Device Architecture
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
WADL	Web Application Description Language
WC3	World Wide Web Consortium

WCI	Wireless Compliance Institute
WEB-DDS	Web Enabled DDS
WG	Working Group
WSN	Wireless Sensory Nodes
XEP	XMPP Extension Protocol
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

4 Conventions,

The key words “Shall”, ”Shall not”, “May”, ”Need not”, “Should”, ”Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 M2M Related Protocols Overview

5.1 Analysis of Design Styles

5.1.1 RESTful Style protocols

The REST architectural style was developed by W3C Technical Architecture Group (TAG) in parallel with HTTP/1.1, based on the existing design of HTTP/1.0.

REST-style architectures conventionally consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that may be used the next time the client chooses to initiate a new state-transition

5.1.1.1 REST Style

The REST architectural style describes the following six constraints applied to the architecture, while leaving the implementation of the individual components free to design:

Client–server: A uniform interface separates clients from servers. This separation of concerns means that, for example, clients are not concerned with data storage, which remains internal to each server, so that the portability of client code is improved. Servers are not concerned with the user interface or user state, so that servers can be simpler and more scalable. Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

Stateless: The client–server communication is further constrained by no client context being stored on the server between requests. Each request from any client contains all of the information necessary to service the request, and any session state is held in the client.

Cacheable: As on the World Wide Web, clients can cache responses. Responses must therefore, implicitly or explicitly, define themselves as cacheable, or not, to prevent clients reusing stale or inappropriate data in response to further requests. Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance.

Layered system: A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers may improve system scalability by enabling load-balancing and by providing shared caches. They may also enforce security policies.

Code on demand (optional): Servers can temporarily extend or customize the functionality of a client by the transfer of executable code. Examples of this may include compiled components such as Java applets and client-side scripts such as JavaScript.

Uniform interface: The uniform interface implies that the interactions between the components in a RESTful architectural style depend on the uniformity of its interface. If any of the components do not follow these constraints, then a RESTful architectural system can result in faults.

The components in a RESTful architectural style interoperate consistently in accordance with the uniform interface's four constraints as follow:

1. Identification of resources: a resource is addressed by a unique identifier, such as URI. (e.g., http://onem2m.com/cse1/application)
2. Manipulation of resources through representations: Clients manipulate representation of resources. The same exact resource can be represented to different clients in different ways. For example, a resource can be represented as HTML or as JSON.
3. Self-descriptive message: A resource's *desired* state can be represented within a request message. A resource's *current* state may be represented within a response message.
4. Hypermedia as the engine of application state (HATEOAS): a resource's state representation includes links to related resources.

The only optional constraint of REST architecture is "code on demand". One can characterise applications conforming to the REST constraints described in this clause as "RESTful". If a service violates any of the required constraints, it cannot be considered RESTful.

5.1.1.2 RESTful Protocols

The following protocols adhere to the principles of RESTful design:

- HTTP as RESTful API
- CoAP

5.2 Data Description Standards

5.2.1 XML Schema Definition Language (XSD)

The XML Schema Definition Language (XSD) can be used to express a set of rules for validating grammatical syntax of XML documents which is handled as specific data types.

XSD 1.0 [i.93] was published as a W3C recommendation in May 2001, and revised XSD 1.1 [i.94] was published in April 2012.

XSD can be automatically referred by XML parsers or validators to check compliance of given XML document.

6 Analysis of Protocols

This clause includes an analysis of protocols relevant to oneM2M including: background, status (current release, under development), category and architectural style, intended use, deployment trend, key features, protocols stack, data model, security aspects, dependencies, benefits and constraints, and support of oneM2M requirements.

6.1 CoAP - Constrained Application Protocol

The following clauses describe the Constrained Application Protocol CoAP. [i.4]

6.1.1 Background

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of

ROM and RAM, while constrained networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of Kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments.

One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other machine-to-machine (M2M) applications. The goal of CoAP is not to blindly compress HTTP [i.7], but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoAP could be used for refashioning simple HTTP interfaces into a more compact protocol, it more importantly also offers features for M2M such as built-in discovery, multicast support and asynchronous message exchanges.

CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity. Multicast, low overhead, and simplicity are extremely important for M2M devices, which tend to be deeply embedded and have much less memory and power supply than traditional internet devices have.

CoAP can run on most devices that support UDP or a UDP analogue, and is intended to be used for M2M / IoT segments such as home building automation and smart metering.

6.1.2 Status

6.1.2.1 Current Status

The IETF Constrained RESTful environments (CORE) Working Group has done the major standardization work for this protocol. In order to make the protocol suitable to IoT and M2M applications, various new functionalities have been added. The protocol has completed IETF last call and is in the final stages of processing for Internet Standards documents.

CoAP is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. CoAP is an application layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensory network (NSN) nodes.

6.1.2.2 Ongoing IETF Activity

CoAP is being standardized with in the IETF CORE working group. Key IETF CoRE WG documents are as follows:

- CoRE Link Format – RFC6690 [i8]
- CoAP Protocol [i.4] – With IESG for publication as an RFC
- Blockwise transfer in CoAP: IETF draft. WG document.
- Observing resources in CoAP – IETF draft. WG document.
- CoRE Resource Directory – IETF draft. WG document
- Group communication for CoAP – IETF draft. WG document.
- Best practices for HTTP to CoAP Mapping Implementation – IETF draft. WG document.

There are several options proposed for use with CoAP. Some of these are as follows:

- Conditional observe in CoAP: IETF draft
- CoAP Patience option: IETF draft
- Enhanced sleep mode support of IoT / M2M devices: IETF draft
- Stateful observation in CoAP (to optimize re-registration traffic in the network): IETF draft
- Minimum request interval for successive CoAP requests – IETF draft

- Transport of CoAP over SMS – IETF draft
- TCP transport for CoAP – IETF draft
- CoAP option to indicate payload length – IETF draft
- CoAP over SMS – IETF draft

6.1.3 Category and Architectural Style

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of Kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments.

The use of web services (web APIs) on the Internet has become ubiquitous in most applications, and depends on the fundamental Representational State Transfer [REST] architecture of the web. The Constrained RESTful Environments (CoRE) work aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN [i.9]). Constrained networks such as 6LoWPAN support the fragmentation of IPv6 packets into small link- layer frames, however incurring significant reduction in packet delivery probability. One design goal of CoAP has been to keep message overhead small, thus limiting the need for fragmentation.

One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other machine-to-machine (M2M) applications. The goal of CoAP is not to blindly compress HTTP [i.7], but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoAP could be used for refashioning simple HTTP interfaces into a more compact protocol, it more importantly also offers features for M2M such as built-in discovery, multicast support and asynchronous message exchanges.

The protocol supports the caching of responses in order to efficiently fulfil requests. Simple caching is enabled using freshness and validity information carried with CoAP responses. A cache could be located in an endpoint or an intermediary.

Proxying is useful in constrained networks for several reasons, including network traffic limiting, to improve performance, to access resources of sleeping devices or for security reasons. The proxying of requests on behalf of another CoAP endpoint is supported in the protocol. When using a proxy, the URI of the resource to request is included in the request, while the destination IP address is set to the address of the proxy.

As CoAP was designed according to the REST architecture [REST] and thus exhibits functionality similar to that of the HTTP protocol, it is quite straightforward to map from CoAP to HTTP and from HTTP to CoAP. Such a mapping may be used to realize an HTTP REST interface using CoAP, or for converting between HTTP and CoAP. This conversion can be carried out by a cross-protocol proxy ("cross-proxy"), which converts the method or response code, media type, and options to the corresponding HTTP feature.

6.1.4 Intended use

CoAP (Constrained Application Protocol) over UDP is used for resource constrained, low-power sensors and devices connected via lossy networks, especially when there is a high number of sensors and devices within the network. Soon to be released as a suite of IETF RFCs, CoAP has already found success as a key enabling technology for electric utility AMI (advanced metering infrastructure) and DI (distributed intelligence) applications

CoAP makes use of two message types, requests and responses, using a simple binary base header format. The base header may be followed by options in an optimized Type-Length-Value format. CoAP is by default bound to UDP and optionally to DTLS, providing a high level of communications security.

6.1.5 Deployment Trend

A First IoT CoAP Plugtest interoperability event organized by ETSI, the IPSO Alliance, and the Probe-IT project was held in March 2012. This interoperability event tested features that included the base CoAP specification, CoAP Block Transfer, CoAP Observation and the CoRE Link Format. As described in draft-bormann-core-roadmap-03, it was attended by 18 companies and more than 3000 tests were performed during this event. The 2nd CoAP plugtest event was held in November 2012.

6.1.6 Key features

The key features of CoAP are:

- CoAP is a RESTful protocol.
- Four methods similar to HTTP: Get, Put, Post and Delete.
- Three types of response code: 2.xx (success), 4.xx (client error) and 5.xx (server error).
- Four different message types: Confirmable, Non-Confirmable, Acknowledgement and Reset (Nack).
- Synchronous message exchange
- Asynchronous message exchange via Observe / Notifications Client uses Observe option with Get request to indicate interest in getting further updates from server. Client receives an asynchronous notification each time state of resource changes at server.
- Conditional Observe allows CoAP clients to be informed only when certain conditions on observed resources are met (such as inform periodically or only inform when observed value changes by a pre-specified step size)
- East to proxy to and from HTTP.
- Constrained web protocol fulfilling M2M requirements.
- UDP [i.10] binding with optional reliability supporting unicast and multicast requests. Confirmable and Acknowledgement / Reset messages to provide optional reliability when required. Asynchronous message exchanges.
- Low header overhead and parsing complexity.
- URI and Content-type support.
- Simple proxy and caching capabilities.
- A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.
- Security binding to Datagram Transport Layer Security (DTLS) (RFC6347 [i.11]) A wide variety of key management mechanisms may be used for this purpose.
- CoRE link format that defines use of Web Linking using link formats for use by constrained devices to describe hosted resources, their attributes and relationships between links (RFC6690 [i.8]) A well-known URI "./well-known/core" is used as a default entry point for requesting list of links about the resources hosted by a server
- A Resource Directory mechanism where IoT / M2M devices (i.e. CoAP servers) can register / update their list of resources. It stores the URIs (called links) to resources stored on servers. If a device is in sleep mode and not able to communicate with the network, it can be discovered via this resource directory. It could also be used if network doesn't support multicast traffic efficiently.
- Mechanism to transfer multiple blocks of information from a resource representation in multiple request-response pairs at CoAP message level itself (i.e. without relying on IP fragmentation). Large file transfers (such as firmware updates) can be done using this mechanism.
- Group based communication using (unreliable) IP multicast by source sending non-confirmable CoAP message to a multicast IP address (and via serial unicast for links that do not support multicast). Some other group based communication mechanisms being explored include the following: overlay multicast that uses proxies to

deliver IP packets to end devices, and support of multicast at CoAP level without any explicit multicast support from lower layers.

- CoAP Patience option that informs a recipient of the preferred time frame for a response or request depending on usage context. Useful for time (or delay) tolerant exchanges
- Proposal for a mechanism where device can register its sleep state and related parameters (such as sleep duration, sleep / active state etc.) with the Resource Directory
- Stateful Observation intends to reduce overhead in the network due to multiple re-registration requests from CoAP client to CoAP server when server (i.e. IoT / M2M device) is not in a position to accept additional clients
- Identification of Proxies between a CoAP client and CoAP server.
- Provides mechanism where a client and server can negotiate the minimum time between two subsequent requests. Helps to reduce excessive load at the CoAP server
- An IETF draft "CoAP Payload Length Option Extension" defines a way to indicate length of the payload when underlying transport layer (such as for RS 232, RS 422 or RS 485) doesn't indicate payload length.
- Mechanism to transport CoAP over SMS for cellular networks
- Representation of links in JSON format in unconstrained environment

6.1.7 Protocol Stack

The interaction model of CoAP is similar to the client/server model of HTTP. However, machine-to-machine interactions typically result in a CoAP implementation acting in both client and server roles. A CoAP request is equivalent to that of HTTP, and is sent by a client to request an action (using a method code) on a resource (identified by a URI) on a server. The server then sends a response with a response code; this response may include a resource representation.

Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP. This is done logically using a layer of messages that supports optional reliability (with exponential back-off). CoAP defines four types of messages: Confirmable, Non-confirmable, Acknowledgement, Reset; method codes and response codes included in some of these messages make them carry requests or responses. The basic exchanges of the four types of messages are somewhat orthogonal to the request/response interactions; requests can be carried in Confirmable and Non- confirmable messages, and responses can be carried in these as well as piggy-backed in Acknowledgement messages. One could think of CoAP logically as using a two-layer approach, a CoAP messaging layer used to deal with UDP and the asynchronous nature of the interactions, and the request/response interactions using Method and Response codes (see Figure 6.1 below). CoAP is however a single protocol, with messaging and request/response just features of the CoAP header.

```
+----------------------+
|      Application     |
+----------------------+
+----------------------+ \
| Requests/Responses | |
|----------------------| | CoAP
|       Messages       | |
+----------------------+ /
+----------------------+
|          UDP         |
+----------------------+
```

Fig. 6.1. Abstract layering of CoAP

6.1.8 Data Model

CoAP allows to explicitly indicate payload of the content type in its header. CoAP Content Format Registry provides following initial entries: plain text, XML, JSON, EXI, octet stream, link-format. New Internet media types may be used depending on the target IoT segment

6.1.9 Security

As CoAP realizes a subset of the features in HTTP/1.1, the security considerations of RFC2616 [i.7] are also pertinent to CoAP. This clause analyses the possible threats to the protocol. There are a number of security limitations with CoAP, and this clause will describe those in detail. These will include:

- Protocol Parsing, Processing URIs
- Proxying and Caching
- Risk of amplification
- IP Address Spoofing Attacks
- Cross-Protocol Attacks
- Constrained node considerations

COAP uses DTLS1.2 and security keys generated by DTLS are used to protect CoAP level messages. Some constraints associated with DTLS are as follows:

- It may be challenging to support DTLS in constrained M2M devices that have limited memory (such as RAM ~ 10 KB) and processing power. This is the reason for the current IETF initiative "DTLS In Constrained Environments" (DICE) initiative (http://www.ietf.org/proceedings/87/slides/slides-87-dice-0)
- Use of DTLS (handshake protocol) results in high overhead in the network and that may not be desirable.
- No clear standardized definition of a constrained DTLS profile

  No efficient support of multicast with IP DTLS. The multicast suitability of CoAP are lost when using DTLS (point-to-point). On this aspect, there are also initiatives attempting to find solutions, e.g. "DTLS-based multicast security for Low power Lossy Networks" (http://tools.ietf.org/id/draft-keoh-tls-multicast-security-00.txt)
- No standardized approaches for (dynamic) key management for group based communication

6.1.10 Dependencies

CoAP is designed to run over datagram transport protocol such as UDP. In this case, it uses DTLS to provide application layer security.

An IETF draft "A TCP transport for CoAP" is exploring changes needed to run CoAP over TCP. Use of CoAP over RS 232 / 422 / 485 is also being explored.

6.1.11 Benefits and Constraints

6.1.11.1 Benefits

CoAP is a lightweight application layer protocol designed for constrained devices (such as devices with 8-bit microcontroller and limited memory) and constrained networks (such as low power, low data rate, lossy networks that use IEEE802.15.4)

- It runs over UDP and avoids overhead of TCP
- It is easy to do HTTP – CoAP translation

6.2.11.2 Constraints

CoAP has:

- Constraints associated with DTLS (as listed in the Security subclause)
- No standardized framework for authorization and access control for CoAP exists as of now. The IETF draft "Access Control Framework for constrained environments" (http://datatracker.ietf.org/doc/draft-selander-core-access-control/?include_text=1) attempts to resolve this issue.
- No explicit support for real-time IoT application at present.

6.1.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by CoAP is shown in the following clauses:

NOTE: Many requirements from TS-0002 [i.2] depend on the architecture of overall M2M system and CoAP comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.1.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-008, OSR-009, OSR-010, OSR-014, OSR-21, OSR-24, OSR-25, OSR-28, OSR-30, OSR-37, SER-002, SER-003, SER-009, NFR-002.

6.1.12.2 Partially Supported Requirements

OSR-004, OSR-005, OSR-006, OSR-007, OSR-011, OSR-013, OSR-016, OSR-017, OSR-018, OSR-019, OSR-020, OSR-021, OSR-022, OSR-027, OSR-029, OSR-030, OSR-031, OSR-032, OSR-033, OSR-034, OSR-035, OSR-036, OSR-037, OSR-038, OSR-039, , OSR-040, OSR-041, OSR-042, OSR-043, OSR-044, OSR-045, OSR-047, OSR-048, OSR-049, OSR-051, OSR-052, OSR-055, OSR-057, OSR-058, OSR-061, OSR-063, OSR-064, OSR-068, OSR-069, OSR-072, SER-008
OSR-060 *(depends on other components in an M2M system where CoAP is being used. For example, one could provide synchronization via IEEE1588v2)*

6.2 MQTT - Message Queuing Telemetry Transport

The following clauses describe the OASIS Message Queuing Telemetry Transport - MQTT protocol. [i.92]

6.2.1 Background

MQTT was invented by IBM and Arcom (now Eurotech), in 1999. It was designed for low-bandwidth, high latency networks. As a result, the designers made a number of key choices which influenced the way it "looks and feels".

1. Simplicity, simplicity, simplicity! Don't add too many "bells and whistles" but provide a solid building block which can easily be integrated into other solutions. Be simple to implement.
2. Publish/subscribe messaging. Useful for most sensor applications, and enables devices to come online and publish "stuff" that hasn't been previously known about or predefined.
3. Zero administration (or as close as possible). Behave sensibly in response to unexpected actions and enable applications to "just work" e.g. dynamically create topics when needed.
4. Minimise the on-the-wire footprint. Add an absolute minimum of data overhead to any message. Be lightweight and bandwidth efficient.
5. Expect and cater for frequent network disruption (for low bandwidth, high latency, unreliable, high cost-to-run networks)… → Last Will and Testament

6. Continuous session awareness → Last Will and Testament
7. Expect that client applications may have very limited processing resources available.
8. Provide traditional messaging qualities of service where the environment allows. Provide "quality of service"
9. Data agnostic. Don't mandate content formats, remain flexible.

MQTT v3.1 was published by IBM in Aug 2010 under a royalty free license. Further pre-OASIS information is available at MQTT.org.

6.2.2 Status

Based on the pre-standard MQTT v3.1 specifications [i.3] there is an OASIS standardization process which started in March 2013 to make MQTT an open, simple and lightweight standard protocol for M2M telemetry data communication. The target for completion is 3rd quarter of 2014 to become an approved OASIS standard.

The OASIS MQTT TC is producing a standard for the Message Queuing Telemetry Transport Protocol compatible with MQTT V3.1, together with requirements for enhancements, documented usage examples, best practices, and guidance for use of MQTT topics with commonly available registry and discovery mechanisms. It operates under the Non-Assertion Mode of the OASIS IPR Policy. Changes to the input document, other than editorial changes and other points of clarification, will be limited to the Connect command, and should be backward compatible with implementations of previous versions of the specification such that a client coded to speak an older version of the protocol will be able to connect to, and successfully use, a server that implements a newer version of the protocol. As of 18 May 2014 the TC approved and published MQTT Version 3.1.1.

The Eclipse foundation through their IoT working group, is providing open source MQTT client libraries via their Paho Project. An Open Source server implementation is being developed by the Eclipse Mosquitto project.

6.2.3 Category and Architectural Style

MQTT is an M2M/Internet of Things (IoT) connectivity protocol. It is connection session reliant. It supports 14 command messages; the message format includes a fixed and variable header plus the payload.

The grouped commands are:

- Client requests a connection to a server, Server acknowledges connection request & Client requests disconnection
- Publish message & Publish acknowledgment
- Assured publish received (part 1), Assured publish release (part 2) & Assured publish complete (part 3)
- Subscribe to named topics & Subscription acknowledgement
- Unsubscribe from named topics & Unsubscribe acknowledgment
- Ping request & Ping response

6.2.4 Intended use

MQTT is designed to support messaging transport from remote locations/devices involving small code footprints (e.g., 8-bit, 256KB ram controllers), low power, low bandwidth, high-cost connections, high latency, variable availability, and negotiated delivery guarantees. For example, MQTT is being used in sensors communicating to a server / broker via satellite links, SCADA, over occasional dial-up connections with healthcare providers (medical devices), and in a range of home automation and small device scenarios. MQTT is also a fit for mobile applications because of its small size, minimized data packets, and efficient distribution of information to one or many receivers (subscribers).

6.2.5 Deployment Trend

MQTT is estimated to be running on 250k devices. It is deployed in the Healthcare Industry Segment (hospitals use the protocol to communicate with pacemakers and other medical devices) and in the Energy Industry Segment (oil and gas companies use MQTT to monitor thousands of miles of oil pipelines). It is also used in Facebook's Messenger application.

MQTT is not deployed in the largest message queue-based telemetry projects.

6.2.6 Key features

The key features of MQTT are:

- Publish/Subscribe - to provide one-to-many message distribution and decoupling of applications
- Topics/Subscriptions – to categorise messages into channels for delivery to subscribers
- Quality of Service – to provide different assurances of message delivery
- Retained messages – to provide past published messages to new subscribers
- Clean session / Durable connections – to choose whether a client's state is to be stored between connection sessions
- Wills – to send messages after a client disconnects unexpectedly

6.2.6.1 Publish/Subscribe

The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a server / broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. Many clients may subscribe to the same topics. The server / broker and MQTT act as a simple, common interface for clients to connect to. A publisher may publish a message once and be received by multiple subscribers.

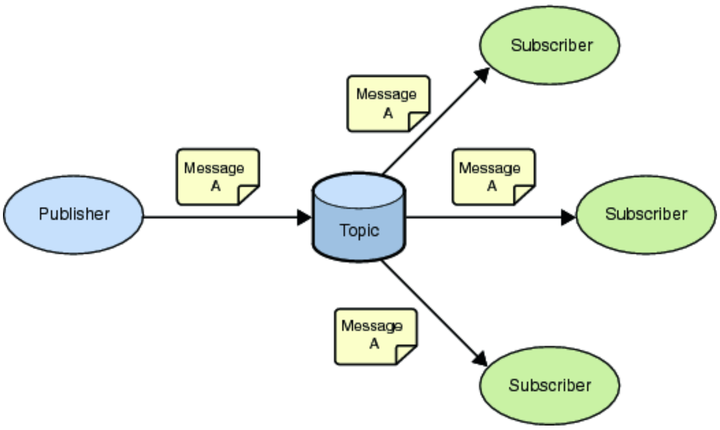


Figure 6.2 MQTT publish-subscribe messaging

6.2.6.2 Topics/Subscriptions

Messages in MQTT are published on topics. Topics are structured into topic trees, which are treated as hierarchies, using a forward slash (/) as a separator. This allows arrangement of common themes to be created. Topics and topic trees can be created administratively, although its more common for a server to create a topic on-demand (subject to security policies) when a client first attempts to publish or subscribe to it.

A subscription may contain special characters, which allow clients to subscribe to multiple topics at once, within a single level or within multiple levels in a topic tree.

6.2.6.3 Quality of Service

MQTT defines three levels of Quality of Service (QoS). The QoS defines how hard the broker & client will try to ensure that a message is received. Messages may be sent at any QoS level, and clients may attempt to subscribe to topics at any QoS level. This means that the client chooses the maximum QoS it will receive. For example, if a message is published at QoS 2 and a client is subscribed with QoS 0, the message will be delivered to that client with QoS 0. If a second client is also subscribed to the same topic, but with QoS 2, then it will receive the same message but with QoS 2. For a second example, if a client is subscribed with QoS 2 and a message is published on QoS 0, the client will receive it on QoS 0.

Higher levels of QoS are more reliable, but involve higher latency and have higher bandwidth requirements.

0. The server / broker & client will deliver the message once, according to the best efforts of the underlying TCP/IP network, with no confirmation. The message arrives at the server either once or not at all.
1. The server / broker & client will deliver the message at least once, with confirmation required. If there is an identified failure of either the communications link or the sending device, or the acknowledgement message is not received after a specified period of time, the sender resends the message
2. The server / broker & client will deliver the message exactly once by using additional protocol flows.

6.2.6.4 Retained Messages

Publish messages may be set to be retained. This means that the server / broker will keep the message even after sending it to all current subscribers. If a new subscription is made that matches the topic of the retained message, then the message will be sent to the client. This is useful as a "last known good" mechanism. If a topic is only updated infrequently (such as for “report by exception”), then without a retained message, a newly subscribed client may have to wait a long time to receive an update. With a retained message, the client will receive an instant update.

6.2.6.5 Durable and non-Durable sessions

MQTT clients choose whether to use durable sessions or not. If a cleints requests a clean connection then a non-durable session is created. The server / broker discards any previously maintained information about the client, the client needs to re-subscribe to topics of interest, and the server / broker discards any state when the client disconnects.

If it does not request a clean session a durable session is used. When the client disconnects any subscriptions it has will remain and any subsequent QoS 1 or QoS 2 messages will be stored until it connects again.

6.2.6.6 Wills

When a client connects to a broker, it may inform the broker that it has a Will. This is a message that it wishes the broker to send to interested parties when the client disconnects abnormally. The Will message has a topic, QoS and retain status just the same as any other message. Abnormal disconnection occurs when either an I/O error is encountered by the server / broker during communication with the client, or the client fails to communicate within the Keep Alive timer schedule.

6.2.7 Protocol Stack

MQTT requires an underlying network protocol that provides ordered, lossless, bi-directional connections, but the MQTT specification does not mandate a particular underlying protocol. In practice implementations use one or more of the following:

- Raw TCP/IP
- TCP/IP with Transport Level Security (TLS)
- WebSocket – either with or without the use of TLS

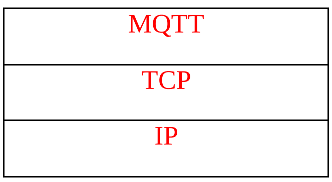


Figure 6.2.7 MQTT Protocol Stack

6.2.8 Data Model

No formal data model has been published. The data elements included in the version 3.1 specification include: topic trees, user name and password, connection state, subscriptions, retained messages, and message headers (fixed and variable).

6.2.9 Security

MQTT supports user names and passwords in connection requests. Connections can be refused due to a bad user name or password.

6.2.10 Dependencies

MQTT uses the TCP/IP layer to provide basic network connectivity.

6.2.11 Benefits and Constraints

6.2.11.1 Benefits

- Protocol compressed into bit-wise headers and variable length fields. Typical message header size is 6 bytes.
- MQTT has been implemented in devices with less than 64kb of RAM
- In comparison to HTTPS, MQTT tested faster throughput, required less battery, and less network overhead.

6.2.11.2 Constraints

- MQTT does not support comprehensive access control. MQTT does support authorization based on a single username & password credential
- MQTT does not define a standard way of fragmenting application-level messages. Applications that need to transmit large messages to constrained memory devices must come up with their own fragmentation scheme.
- MQTT does not support transactions; there is no way to rollback a message once it has been sent, and no way of grouping a batch of separate messages into a single unit-of-work.
- MQTT does not explicitly address connection security; it relies on the transport layer to provide an appropriate level of integrity and encryption.
- MQTT does not support discovery of clients or servers
- MQTT is not extensible, requiring a new protocol revision to evolve capabilities

6.2.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by MQTT is shown in the following clauses: TLS is highly recommended to be used with MQTT but is not considered in the following clauses considering requirements.

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and MQTT comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.2.12.1 Fully Supported Requirements

The following oneM2M requirements [i.2] are fully supported by MQTT [i.92]:

OSR-001, OSR-003, OSR-008, OSR-009, OSR-012, OSR-015, OSR-016, OSR-018, OSR-019, OSR-020, OSR-021, OSR-025, OSR-028, OSR-029, OSR-030, OSR-046, OSR-047, OSR-049, OSR-55, OSR-065, ABR-001, SER-002, SER-008, SER-011, SER-019, SER-025, SER-026, CPR-002, CPR-005.

6.2.12.2 Partially Supported Requirements

The following oneM2M requirements [i.2] are partially supported by MQTT [i.92]:

OSR-002, OSR-010, OSR-017, OSR-022, OSR-024, OSR-041, OSR-044, OSR-045, OSR-48, OSR-053, OSR-059, OSR-067, SER-007, SER-009, SER-017, SER-018.

6.2.12.3 Unsupported Requirements

The following oneM2M requirements [i.2] are not supported by MQTT [i.92]:

OSR-004, OSR-005, OSR-006, OSR-007, OSR-011, OSR-013, OSR-014, OSR-023, OSR-026, OSR-027, OSR-031, OSR-032, OSR-033, OSR-034, OSR-035, OSR-036, OSR-037, OSR-038, OSR-039, OSR-040, OSR-042, OSR-043, OSR-050, OSR-051, OSR-052, OSR-054, OSR-056, OSR-057, OSR-058, OSR-060, OSR-061, OSR-062, OSR-063, OSR-064, OSR-066, OSR-68, OSR-69, OSR-70, OSR-071, OSR-072, MGR-001, MGR-002, MGR-003, MGR-004, MGR-005, MGR-006, MGR-007, MGR-008, MGR-009, MGR-010, MGR-011, MGR-012, MGR-013, MGR-014, MGR-016, MGR-017, ABR-002, ABR-003, SMR-001, SMR-002, SMR-003, SMR-004, SMR-005, SMR-006, SMR-007, SER-001, SER-003, SER-004, SER-005, SER-006, SER-010, SER-012, SER-013, SER-014, SER-015, SER-016, SER-020, SER-021, SER-022,SER-023, SER-024, CHG-001, CHG-002, CHG-003, CHG-004, CHG-005, CHG-006, OPR-001, OPR-002, OPR-003, OPR-004, OPR-005, OPR-006, CRPR-001, CRPR-003, CRPR-004.

6.3 TIA TR-50 Protocol

6.3.1 Background

TR50 Protocol has been designed for easy to use. The protocol is based on a simple architecture that emphases on the connection between the entities rather than forcing a specific architecture.

The intent of the protocol was to be a binding protocol not a transport protocol. In the current architecture of the protocol, the interconnectivity between different nodes can be achieved in two way:

1. Direct connection – in this case devices and applications (node) connect one to each other directly.
2. Using a hub and spoke configuration – in this case devices and applications (nodes) connect to a broker.

In both case, the connections are established using a transport protocol (e.g. MQTT, HTTP) and can use TLS to secure the communication. The TR50 protocol is using these, established, connections to exchange information. The transport protocol incorporates the TR50 protocol in the payload. The TR50 is based on JSON format for easy to understand and debug during the implementation.

6.3.2 Status

The TIA TR-50 Protocol [i.5] has been published by the TIA TR-50 Smart Device Communication engineering committee in December 2012.

6.3.3 Category and Architectural Style

TR-50 is an M2M binding protocol. The protocol is session oriented and authentication credentials must be supplied every time a command is executed.

The protocol consist of 2 grouped commands are:

- Basic Commands
- Security Commands

The architecture includes:

- One or more host nodes
- One or more intermediate nodes
- One or more devices
- An Authentication/Authorization/Accounting node

6.3.4 Intended use

The protocol is designed to be used in conjunction with one of the common publisher/subscriber transport protocol like MQTT or it can be used in conjunction with a simple transport protocol like HTTP. Since the protocol is based on JSON, it can be used with any session oriented transport protocols.

6.3.5 Deployment Trend

The TR-50 protocol is the key feature for horizontal platforms delivering powerful, scalable and reliable architectures.

6.3.6 Key features

The TR-50 key features are:

- Flexible to the use of JSON format
- Extensible – commands/methods can be added on the core protocol or can be added specific per implementation
- Flexible architecture – protocol takes advantage of the flexible architecture to allow simple or complex implementations.
- Protocol can used any data format due to the on-demand definitions.
- Text-based protocol that can be implemented by simple devices as well as complex applications.

6.3.7 Protocol Stack

The TIA TR-50 Protocol is based on a frame that is flexible to provide extensions for further enhancements and/or improvements.

6.3.7.1 Frame Details

The communication is based on:

- Requests
- Responses

6.3.7.2 Request Frame Details

The M2M request frames are based on a JSON structure and consist of two major clauses:

- Authentication – this is the place where the entire authentication items are placed.
- Command(s) – this is the place where the commands items are placed

Below is the structure of the oneM2M frame:

```
{
  "auth": {
    "applicationToken": "<application token>",
```

```
    "sessionId": "<session token>"
  },
  "ref": {
    "command": "<command keyword>",
    "params": []
              }
              }
```

In order to minimize the traffic, it is possible to have M2M frames with multiple commands:

```
"ref1": {
    "command": "<first command>",
    "params": []
},
"ref2": {
    "command": "<second command>",
    "params": []
}
```

The description of the fields is presented in Table 6.3.1 below

Name	Description	Type	Mandatory
auth	Keyword. Identifies the authentication stanza in the M2M request frame	String	Yes
applicationToken	Keyword. Identifies the application making the request	String	Yes
sessionId	Keyword. Unique ID received after the use of the authentication services. Identifies the current session between the two entities	String	Yes
ref1, ref2, ref3	Identifies the commands that are in the request. Can be simple identifiers. They are not keyword. It is expected that the response will contain them.	String	Yes
command	Keyword. Identifies a known command that can be executed	String	Yes
params	Keyword. Identifies the parameters required by the command. The field must exist, but it can be empty.	String	Yes

Table 6.3.1 Request Frame Field Descriptions

6.3.7.3 Response Frame Details

The response frame is based on the JSON format and it has the following structure:

```
{
  "ref1": {
    "success": true,
```

```
        "params": []
      },
      "ref2": {
        "success": false,
      "errorCodes":
        [
          <errorCode1>,
          <errorCode2>
        ]
      "errorMessages":
        [
          <errorMessage>,
          <errorMessage>
        ]
      },
    }
```

- “ref1” response is an example for a positive return of a request
- “ref2” response is an example for an negative or error return of a request

The description of the fields is presented Table 6.3.2 below

Name	Description	Type	Mandatory
ref1,ref2, ref3	Identifies the request commands.	String	Yes
success	Keyword. Identifies the response. Can be true or false	String	Yes
errorCodes	Keyword. Identifies the error codes .		
errorMessages	Keyword. Identifies the error message.	String	No
params	Keyword. Identifies the response parameters. Can be empty.	String	Yes

Table 6.3.2: Response Frame Field Descriptions

6.3.8 Data Model

Common data has been published which includes data types as:

- INT1, INT2, INT4, INT8, UINT1,
- UINT2, UINT4, UINT8, FLOAT4,
- FLOAT8, STRING, BOOL, BINARY (Base64 Encoded)

6.3.9 Security

The security of the protocol depends one two major factors:

1. The security of the transport protocol
2. The availability of intrinsic security commands/methods like:
    - api.authenticate
    - api.deauthenticate
    - api.authorize
    - api.deauthorize

6.3.10 Dependencies

TR50 Protocol depends on a transport protocols like HTTP or MQTT. The protocol can be used using direct socket communications via TCP or UDP.

The protocol uses the transport protocol to guarantee the delivery of the data between the nodes.

The protocol includes session oriented features to guarantee the proper functionality during the exchange of the information.

6.3.11 Benefits and Constraints

6.3.11.1 Benefits

- Text-based protocol – ability to be compressed over cellular networks
- JSON-based protocol – simple to understand and debug during the development phase
- Extensible – methods / commands can be added on demand
- Based on very loose architecture
- Suitable for publishing / subscribing architectures

6.3.11.2 Constraints

- TR-50 is not a transport protocol
- Dependent on a transport protocol

6.3.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by TR-50 is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and TIA-TR50 TIA-4940-020 [i.5] comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.3.12.1 Fully Supported Requirements

OSR-001,OSR-002,OSR-003,OSR-004, OSR-007,OSR-009,OSR-010, OSR-012,OSR-013,OSR-014,OSR-015,

OSR-017,OSR-019,OSR-020, OSR-021,OSR-022,OSR-023,OSR-024,OSR-025, OSR-027,OSR-029,OSR-030,

OSR-034,OSR-035,OSR-036,OSR-037, OSR-041,OSR-044,OSR-046,OSR-047,OSR-049,OSR-050, OSR-053,

OSR-054,OSR-055,OSR-056,OSR-057,OSR-059, OSR-064,OSR-065,OSR-066,OSR-067, OSR-071,OSR-072

6.3.12.2 Partially Supported Requirements

OSR-005, OSR-006, OSR-018, OSR-026, OSR-069, OSR-016, OSR-032, OSR-070

6.3.12.3 Unsupported Requirements

OSR-011,OSR-062,OSR-063,OSR-069, OSR-031,OSR-043,OSR-033,OSR-040,OSR-045,OSR-048,OSR-060, OSR-061, OSR-038,OSR-039,OSR-042, OSR-051, OSR-052

6.4 HTTP as RESTful API

6.4.1 Description

HTTP protocol is widely used for Web Services in different ways. One of those usages is using HTTP as RESTful API (HTTP-REST-API).

HTTP-REST-API, will provides CRUD (Create/Read/Update/Delete) operational primitives naturally with its method (e.g. POST/GET/PUT/DELETE) and well-defined status codes.

HTTP-REST-API is subset of HTTP with RESTful design style.

HTTP-REST-API usually supports XML or JSON (JavaScript Object Notation) for passing API parameters. HTTP-REST-API can handle various data formats using Content-Negotiation feature which is part of HTTP specification.

6.4.2 HTTP Status

6.4.2.1 HTTP/1.x Status

- HTTP version 1.1 was published as RFC 2616 [i.7] (Draft Standard) in June 1999.
- Extensible Markup Language (XML) 1.0 (Fifth Edition) [i.12] was published as W3C Recommendation on 26 November 2008.
- “The application/json Media Type for JavaScript Object Notation (JSON)” [i.13] was published as RFC4627 on July 2006.

6.4.2.2 HTTP/2.0 (httpbis) Status

The Hypertext Transfer Protocol Bis (httpbis) Working Group of the IETF is working on HTTP/2.0, which is intended to supersede HTTP/1.1. It is expected that HTTP/2.0 will be submitted to IESG for consideration as a Proposed Standard in Nov 2014. HTTP/2.0 is intended to retain the semantics of HTTP without the legacy of HTTP/1.x message framing and syntax, which have been identified as hampering performance and encouraging misuse of the underlying transport. As part of the HTTP/2.0 work, the following issues are being considered:

- A negotiation mechanism that is capable of not only choosing between HTTP/1.x and HTTP/2.x, but also for bindings of HTTP URLs to other transports (for example).
- Header compression (which may encompass header encoding or tokenisation)
- Server push (which may encompass pull or other techniques)

It is expected that HTTP/2.0 will:

- Substantially improve end-user perceived latency in most cases, over HTTP/1.1 using TCP.

- Address the "head of line blocking" problem in HTTP/1.1 created by pipelining
- Not require multiple connections to a server to enable parallelism, thus improving its use of TCP, especially regarding congestion control
- Retain the semantics of HTTP/1.1, including HTTP methods, status codes, URIs, and where appropriate, header fields.
- Define how HTTP/2.0 interacts with HTTP/1.x, (both 2->1 and 1->2).
- Identify any new extensibility points and policy for their appropriate use.

The resulting specification(s) are expected to meet these goals for common existing deployments of HTTP; in particular, Web browsing (desktop and mobile), non-browsers ("HTTP APIs"), Web serving (at a variety of scales), and intermediation (by proxies, corporate firewalls, "reverse" proxies and Content Delivery Networks). Likewise, current and future semantic extensions to HTTP/1.x (e.g., headers, methods, status codes, cache directives) should be supported in HTTP/2.0.

6.4.4 Intended Use

HTTP-REST-API provides easy to understand, scalable, secure APIs for Web service in distributed computing environments, such as the Internet.

6.4.5 Deployment Trend

According to the ‘API Directory’ provided by independent web site ‘programableweb.com’, over 6000 RESTful APIs are published (as of Aug 27th, 2013).

Since API’s value can be enhanced by combined use of other APIs, called ‘mash up’, choosing API to be ‘RESTful’ potentially increase its value as twice or more.

W3C published WADL specification [i.95] to describe HTTP-REST-API. Several tool can generate skeleton codes for web application from WADL definition.

6.4.6 Key Features

6.4.6.1 Relevant Instance of RESTful Design

Since HTTP specification are designed to implement RESTful architecture, you can develop robust, secure, scalable system with HTTP-REST-API.

HTTP-REST-API also can be secured by applying TLS. Unlike other solutions, TLS will not imply the complexity. Additionally, there are many hardware-based solutions to accelerate crypt graphical processing like load balancer, embedded co-processor.

6.4.6.2 Using XML and JSON

Both XML and JSON can carry extensible data structures easily.

Even JSON format can transfer same information in smaller size than XML, XML can provide strong message-level security, like partial encryption and/or digital signature which cannot be provided by JSON.

REST isn't just about JSON or XML though, but any of the media types that the browser or platform can natively handle with content negotiation mechanism which is part of HTTP specification.

6.4.9 Security

HTTP-REST-API is based Web services are prone to the same vulnerabilities as standard web applications, including broken authentication, injection attacks, cross-site scripting and cross-site request forgery.

Fortunately, many HTTP security practices can be successfully applied for securing HTTP-REST-API.

The Web Service with HTTP-REST-API can be secured by those rules with configuring a policy, ensuring that access to the service requires usage of TLS and authorizing service access based on group membership.

6.4.10 Dependencies

HTTP-REST-API depends on HTTP, XML, JSON, and MIME specifications.

6.4.12 Support of oneM2M Requirements

Support of oneM2M Requirements [i.2] by HTTP as RESTful API is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and HTTP RESTful APIs comprise one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.4.12.1 Fully Supported requirements

OSR-001,OSR-002,OSR-003,OSR-004, OSR-007,OSR-009,OSR-010, OSR-012,OSR-013,OSR-014,OSR-015, OSR-017,OSR-019,OSR-020, OSR-021,OSR-022,OSR-023,OSR-024,OSR-025, OSR-027,OSR-029,OSR-030, OSR-034,OSR-035,OSR-036,OSR-037, OSR-041,OSR-044,OSR-046,OSR-047,OSR-049,OSR-050, OSR-053, OSR-054,OSR-055,OSR-056,OSR-057,OSR-059, OSR-064,OSR-065,OSR-066,OSR-067, OSR-071,OSR-072

6.4.12.2 Partially Supported Requirements

OSR-005, OSR-006, OSR-018, OSR-026,
OSR-069: *(Subject to restriction based on Network Operator's policy)*
OSR-016: *(possible to implement with long-polling mechanism)*
OSR-032,
OSR-070: *(there are no standardized ways to Notify, but it could be specified within oneM2M)*

6.4.12.3 Unsupported Requirements

OSR-011,OSR-062,OSR-063,
OSR-069: *(Cannot control feature of underlying network)*
OSR-031,
OSR-043: *(There is no concept of 'group')*
OSR-033,OSR-040,OSR-045,OSR-048,OSR-060,
OSR-061: *(Not applicable)*
OSR-038,OSR-039,
OSR-042: *(There is no concept of 'QoS')*
OSR-051: *(Supports only communication with request and response pairs)*
OSR-052: *(Multicast transport is not supported)*

6.5 XMPP: eXtensible Messaging and Presence Protocol

The following clauses describe the eXtensible Messaging and Presence Protocol (XMPP). [i.6]

6.5.1 Background

XMPP was first proposed by Jabber open source community and later formalized by IETF in RFC3920. It is an open XML-based protocol for near real-time messaging, presence and request-response services. Several extensions have been added to achieve other capabilities.

6.5.2 Status

IETF RFCs and drafts:

- RFC6120: XMPP: Core (Standards Track RFC. Obsoleted RFC3920) [i.6]. It defines the base XMPP protocol along with RFC6121.
- RFC6121: XMPP: Instant Messaging and Presence [i.14]. Standards track RFC (obsoletes RFC 3921)
- RFC6122: XMPP Address Format (Standards Track) [i.15]. Updates RFC3920

XEP (XMPP Extension Protocol) documents specify extensions to XMPP and are standardized by XSF (XMPP Standards Foundation, http://xmpp.org/extensions/).

6.5.3 Category and Architectural Style

XMPP uses a federated client-server model with multiple interconnected servers as shown in Figure 6.5.1. Each server is responsible for managing its own domain and works cooperatively with servers of other domains as peers.

XMPP supports Availability for Concurrent Transactions (ACT) style where asynchronous end-to-end exchange of structured data is carried out using direct and persistent XML streams among a distributed network of globally addressable, presence aware clients and servers (RFC6120 [i.6]).

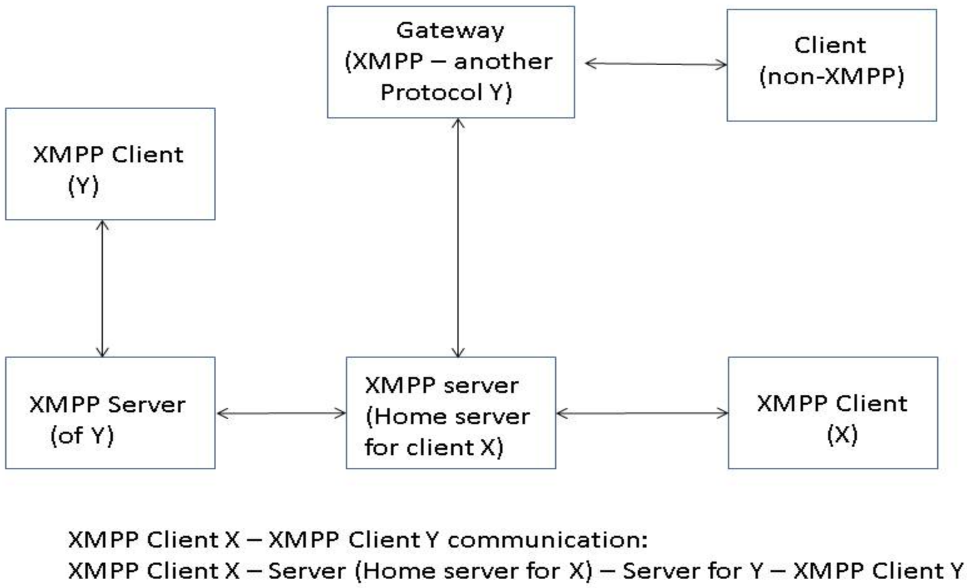


Figure 6.5.1: XMPP Federated Client – Server Model

6.5.4 Intended use

XMPP is intended to be used for P2P, P2M and M2M / IoT purposes.

6.5.5 Deployment Trend

Millions of users world-wide use XMPP for instant messaging and presence based applications.

List of some servers that support XMPP is given at http://xmpp.org/xmpp-software/servers/. These servers provide basic messaging, presence and XML routing features. These servers are available for different platforms such as Linux, Solaris, Windows and Mac OS X.

A list of XMPP clients is available at http://xmpp.org/xmpp-software/clients/. Clients are available for different platforms such as Linux, Windows, Android, Blackberry, iOS, Mac OS X, J2ME, Palm OS and Browser.

A list of XMPP software libraries is available at http://xmpp.org/xmpp-software/libraries/. These libraries are implemented in various languages such as C, C++, Java, Perl, Ruby, PHP, Python, JavaScript, Tcl, Objective C, Flash / Action Script and C # /.Net/Mono.

XMPP is used by the Jabber messaging client. The first instant messaging service based on XMPP was Jabber.org.

- Jabber is used for text conferencing of IETF meetings.
- Cisco / WebEx uses XMPP.
- Google Talk uses XMPP protocol for instant messaging and presence. It uses extensions of XMPP for VoIP, video and peer-to-peer communication.
- Microsoft provides XMPP interface to its Microsoft Messenger Service and have XMPP gateways integrated in their messaging systems.
- Facebook presents an XMPP interface to its clients for its chart feature

6.5.6 Key features

XMPP supports Availability for Concurrent Transactions (ACT) style (RFC6120 [i.6]).

XMPP supports a distributed client server architecture where a client needs to connect with a server to gain access to network. Only after that, it is allowed to exchange XML stanzas with other entities in the network (e.g. in a different domain). End-to-end communication is logically peer-to-peer but physically client-to-server, server-to-server and server-to-client.

TLS (RFC 4492 [i.16]) is supported for encryption purposes (between client – server and server – server).

SASL (Simple Authentication and Security Layer, RFC 4422 [i.17]) is used for authentication of initiating entity (e.g. an XMPP client) with the receiving entity (e.g. XMPP server) before the initiating entity is allowed to send XML stanzas to receiving entity.

Server to server model allows authenticated and secure inter-domain communication.

Each domain is controlled by a server in a decentralized architecture. Each domain owner can define level of security needed, QoS and policies that are needed for that domain.

Relevant XMPP Extension Protocols (XEPs) include:

- XEP-0016 Privacy Lists [i.18]: It can be used to block communication from some XMPP users. It can be potentially used for M2M applications as well. In some sense, it allows to implement simple policies to allow or block access to an M2M device from another M2M device.
- XEP-0030 Service Discovery [i.19]: Allows to discover XMPP entities and features supported by these entities.
- XEP-0045 Multi-user conferencing service. [i.20]
- XEP-0060 Publish-Subscribe service [i.21]. It enables a service to generate notifications and deliver those to multiple subscribers. It is more generalized than the special form of publish-subscribe model supported by presence service. Personal Event Profile (XEP-0163) specifies a stripped down profile of pubSub. Publish-Subscribe feature helps to support asynchronous communication for IoT / M2M applications.
- XEP-0079 Advanced-Message Processing [i.22]. Allows including message expiration feature. It can be useful to indicate expiration time for M2M data that is cached.
- XEP-0080: Allows publishing location information [i.23]; Useful to associate location information with M2M devices.
- XEP-0136 Message Archiving [i.24]: In addition to P2P applications, this can be potentially used for caching M2M data at an XMPP server.
- XEP-0138 – Supports application layer compression [i.25]; Useful in constrained IoT environment.
- XEP-0149 – Allows XMPP entities to specify time period for state, event or activity [i.26].

- Jingle specifications (XEP-0166 [i.27], XEP-0167 [i.28], XEP-0177 [i.29], ….) extend XMPP for initiating and managing peer-to-peer media sessions between two XMPP entities. It uses XMPP for signalling purposes and uses different transport methods (such as TCP, UDP, ….) for data plane packets.
- XEP-0198 Reliability, Stream Management Protocol [i.30]
- XEP-0199 : Provides support for application level pings [i.31]
- XEP-0124 [i.32] and XEP-0206 [i.33]: Allows use of HTTP as transport for XML streams.
- XEP-0203 [i.34]: In addition to P2P applications, this can potentially be also used for M2M applications. It allows an XMPP server to store a message if corresponding XMPP client (e.g. an M2M device) is in offline (e.g. sleep) state and send message as soon as it gets to know that the client / device is available. It gets to know the status via presence notification as the client / device moves from offline (sleep) state to available state.
- XEP-0322 Efficient XML Interchange (EXI) Format for XMPP [i.35]. Useful for M2M applications.
- XEP-0323 Sensor data [i.36]: It provides architecture, data structures and basic operations for sensor (M2M) data communication over XMPP networks. This is designed for implementation in sensors that may have limited amount of memory or processing power.
- XEP-0324 IoT Provisioning [i.37]
- XEP-0325 Internet of Things – Control [i.38]: It provides mechanism to control actuators in XMPP based sensor networks.
- XEP-0326 Internet of Things – Concentrators [i.39]

6.5.7 Protocol Stack

As described in RFC6120 [i.6], XMPP is an application profile of XML that enables near real-time exchange of structured yet extensible data between any two or more network aware entities. An XMPP address (called a Jabber Identifier or JID) is represented as localpart@domainpart/resourcepart. An example of JID is tom@jabber.org/Laptop.

For client-to-server communication, client resolves FQDN of the receiving entity to an IP address and opens a TCP connection to the advertised port at receiver's IP address. Channel encryption is optionally supported using TLS and authentication is done using SASL. After a client authenticates with a server, it binds a specific resource to the stream so that server can properly address the client. Address for use over that stream is a full Jabber ID of the form <localpart@domainpart/resource>. Client opens an XML stream over TCP with a server (of its domain) and exchanges XML stanzas.

Here, an XML stream is a container for carrying XML elements between two elements in a network. XML elements in an XML stream carry XML stanzas (i.e. actual payload message in XML format) or the elements that are used to negotiate the stream (e.g. for TLS and SASL purposes). Each stream is unidirectional and thus two streams are needed between an initiating entity and a receiving entity for bidirectional transfer of XML stanzas. An XML stanza is a basic unit of meaning in XMPP. An XML stream begins with an open tag <stream> and ends with a close tag </stream>. Each direction of conversation is represented as a streaming XML document that ends when that connection is terminated. Root node of that streaming document is the <stream/> element.

For server-to-server communication, a server opens an XML stream over TCP with a server of different domain for inter-server (or inter-domain) communication. Server-to-server streams are typically negotiated in the initialization phase.

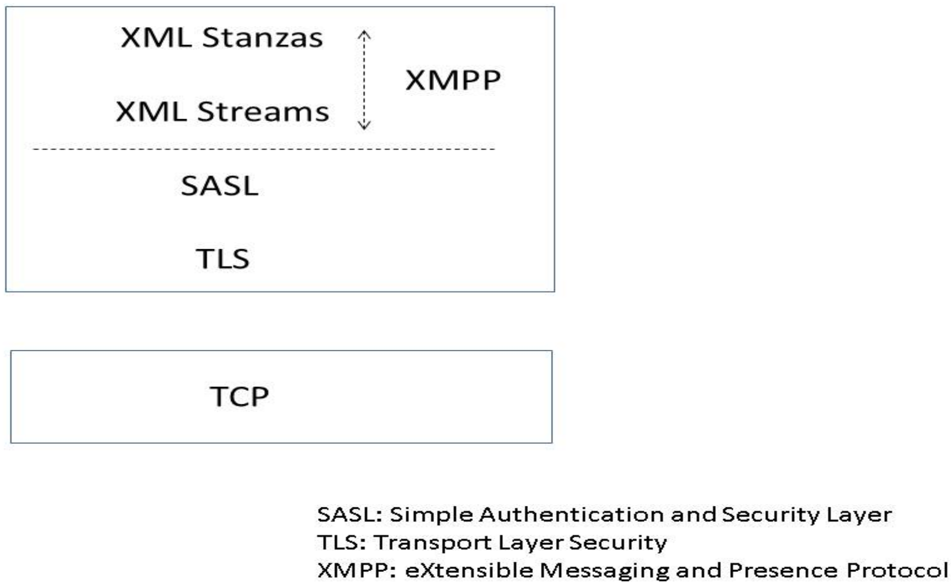


Figure 6.5.2: XMPP Protocol Stack

There are three core stanza types, <presence/>, <message/> and <iq/>, each with its own semantics. These three core stanzas are briefly described below:

- **presence stanza:** A basic publish-subscribe mechanism that allows several entities to receive information (about presence or availability) of a specific entity to which they have subscribed. In an IM application, presence information of a user's (or client's) contacts is displayed in user's contact list or roster. When a user gets online, XMPP software announces user's current status to server of user's domain and that server informs user's contacts about its online status. It is a simple broadcast mechanism in that sense. The server also informs the presence status of user's contacts to user. An M2M device if not available for processing some action can use "do not disturb" to indicate that it is not available.

- **message stanza:** This supports a push mechanism where one entity pushes information to another entity. It supports real-time as well as delayed (i.e. store and push) delivery of messages. In an IM scenario, the message typically encapsulates chat data. Messages are also used for group chat, event delivery and notifications. Message type includes the following: normal, chat, groupchat, headline and error. Message type "headline" is used to send alert and notifications. Type "chat" is used for Instant Messaging.

  For IM, XMPP servers are optimized to handle large number of small messages. As an XMPP server knows about the availability (or presence) status of XMPP clients, it can quickly take an appropriate decision. It can either send message to that client quickly if that client is available or can store it in buffer and send as soon as it gets to know that the client is available. This feature can also be used for M2M purposes.

- **IQ (Info/Query) stanza:** A request-response mechanism that allows structured exchange of data between an initiating entity and a requesting entity in a somewhat reliable way. It allows operations such as get (reading), set (a variable), result and error. Values of type attribute used with IQ stanza are get, result, set and error.

6.5.7.1 XEP-0323 Sensor data

XEP-0323 [i.36] describes framework for sensor data exchange in an XMPP based sensor network (SN). Specific support of XMPP SN feature is indicated by including "urn"xmpp:sn" in the service discovery procedure. A sensor device, an actuator or a gateway is an example of a node in a sensor network. A Concentrator is a device that handles multiple nodes (e.g. a node handling multiple sensors) behind it. Field Name is name of a field of sensor data (e.g. pressure or vibration level). Possible values of Field Type include the following: momentary value, calculated value, peak value, status value, historical value etc.

6.5.7.2 XEP-0324 IoT Provisioning

XEP-0324 [i.37], IoT provisioning, deals with access rights, user privileges and provisioning of services in a sensor network. This architecture uses distributed third parties that provide the following services:

- Control who can communicate with whom (i.e. control friendship)
- Control read access
- Control configure / write access
- Provide a user interface to set / update these policies
- Provide interoperability services (such as unit conversion)

A trust relationship is created between a device and a provisioning server using some mechanisms. A device, client or user can get a token from a provisioning server that it can use to validate access rights with the server. <iq> stanza with xmlns 'urn:xmpp:sn:provisioning' is used for this purpose. If a device supports provisioning feature, it advertises this feature in service discovery.

If there are multiple provisioning servers, device / client / user has one token from each of the provisioning server. While sending request to another entity (e.g. read or write some data), it includes all these tokens. As an IoT device gets a request to read or write some data, it contacts a provisioning server with the tokens provided in the request and validates access rights of the requesting entity (Figure 6.5.3). As an IoT device gets friendship request from a third party, it contacts provisioning server and checks whether or not to accept that request. A provisioning server can also delegate a secondary trust to a device by which that device can add its own friends.

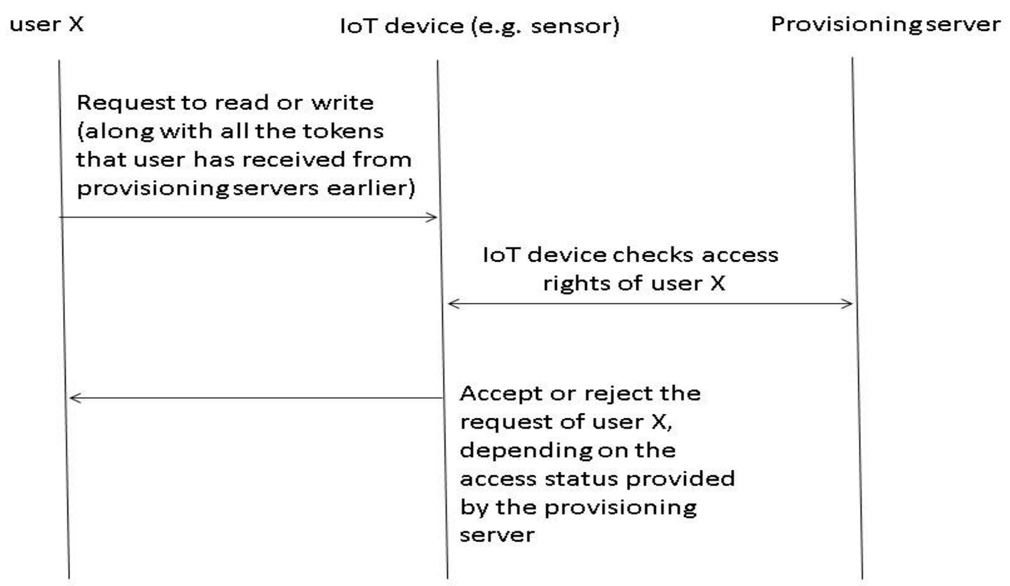


Figure 6.5.3: Validation of access rights during a read or write operation

6.5.7.3 XEP-0325 Internet of Things - Control

XEP-0325 [i.38] specifies mechanisms to control actuators in an XMPP based sensor network. <message> or <iq> stanzas can be used for this purpose. Response from device is suppressed when using <message> stanza but <iq> stanza can be used if an acknowledgement is needed from the actuator. Operation "Set" and the xmlns "urn:xmpp:sn:control" are used for this purpose in the message stanza. A control form can also be used to set values for various fields at the actuator. Actuators behind a concentrator can be controlled by specifying node elements for those actuators.

6.5.7.4 XEP-0326 Internet of Things - Concentrators

XEP-0326 [i.39] deals with IoT Concentrators. A concentrator is defined to be a device that concentrates management of a subset of devices (of a sensor network) at a point. A concentrator can be small (e.g. a PLC managing a set of sensors and actuators), medium (e.g. a branch of a network using a different communication protocol), large (e.g. a sub-system managed by a partner organization) or massive. A concentrator works with multiple data sources where a data source is defined to contain a collection of nodes. There are three types of data sources: Singular, Flat and Tree. There is only one node object in a singular data source while a flat data source contains list of node objects. Nodes are represented in a tree structure in a tree data source. Asynchronous events are sent from a concentrator to each client that has subscribed to these using message stanzas. A concentrator can also store data from sensors locally (or at a remote server but controlled locally).

A client that needs to communicate with a concentrator can get type of commands supported by getting capabilities of the concentrator. The xmlns ‘ ’ is used for this purpose. Some such commands are given below:

- Get all data sources managed by the concentrator,
- Get root data sources,
- Get child data sources (of a root data source),
- Given node id, check to see if a node is supported by a (given) concentrator,
- Get basic information about a node supported by a concentrator (e.g. is it readable? Is it configurable? what are the parameters supported by a node e.g. location of a meter? etc.)
- Change order of nodes in a tree (by moving nodes up or down among siblings),
- Get and set parameters of a node,
- Create or destroy a node,
- Get commands that are supported by a node,
- Subscribe to changes in data source by allowing devices to register for asynchronous events,
- Allow retrieval of historic events

6.5.8 Data Model

XMPP is based on XML. EXI (Efficient XML Interchange) is also supported for M2M applications.

6.5.9 Security

TLS is supported for encryption of client-to-server and server-to-server communication; its use is optional. A receiving entity (such as a client or a peer server) can mandate that the initiating entity use TLS for data encryption.

An initiating entity needs to authenticate with the receiving entity before sending XML stanzas. If TLS is used, it is used before negotiating for SASL (Simple Authentication and Security Layer Protocol). It helps protect the authentication information exchanged during SASL negotiation.

6.5.10 Dependencies

- XMPP streams as defined in RFC6120 [i.6] use TCP as transport
- Use of HTTP as transport is allowed as per XEP-0124 [i.32] and XEP-0206 [i.33]
- Uses TLS and SASL for security.
- Jingle extensions use XMPP for signalling but data plane packets are sent over other transport mechanisms such as TCP, UDP, ….
- XML

6.5.11 Benefits and Constraints

XMPP is an open standard. IETF has approved XMPP RFCs for core methods (RFC6121), instant messaging and presence technology (RFC6121 [i.14]) and address format (RFC6122 [i.15]). XMPP standard foundation and IETF continue to extend this.

6.5.11.1 Benefits

- XMPP is easily extensible. It provides basic set of features that can be expanded by protocol extensions (XEPs) to provide new set of features.
- Resource location is specified in the address itself and that makes it easy to identify different resources of an XMPP user.
- XMPP is already used by some devices for IM applications. In that sense, it improves Person-to-Machine (P2M) communication as user is able to directly interact with smart object running XMPP. It does not necessarily require use of protocol gateways as may be needed with CoAP (e.g. for HTTP to CoAP conversion) and MQTT.
- It supports a federated client-server architecture where no (global) centralized server is needed. Anyone with a domain name can run an XMPP server on its own domain. Public XMPP servers are available for everyone.
- Support of message, presence and IQ stanza types helps meet needs of several IoT applications.
- As it uses “store and push” mechanism to transfer data, it can store contents if the receiving entity is offline (e.g. if IoT device is in sleep mode).
- <xml:lang> common attribute enables internationalization.
- An XMPP address (or JID) can include any Unicode character and is not restricted to ASCII characters.
- Some of the existing mechanisms (such as publish-subscribe, caching, delayed delivery, support for EXI, etc.) are applicable for IoT use cases as well.

6.5.11.2 Constraints

- Use of TCP may not be desirable for some IoT segments.
- Overhead may be high if XML data is used but EXI extensions available for IoT applications.

6.5.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by XMPP is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and XMPP comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.5.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-008, OSR-009, OSR-010, OSR-012, OSR-014, OSR-015, OSR-024, OSR-025, OSR-026, OSR-028, OSR-047, OSR-048, OSR-049, OSR-053, OSR-058, SER-002, SER-003, SER-009, NFR-002.

6.5.12.2 Partially Supported Requirements

OSR-005, OSR-006, OSR-007, OSR-009, OSR-011, OSR-013, OSR-016, OSR-017, OSR-018, OSR-019, OSR-020, OSR-021, OSR-022, OSR-023, OSR-027, OSR-029, OSR-030, OSR-031, OSR-032, OSR-033, OSR-034, OSR-035, OSR-036, OSR-037, OSR-038, OSR-039, OSR-040, OSR-041, OSR-042, OSR-043, OSR-044, OSR-045, OSR-054, OSR-057, OSR-063, OSR-064, SER-008.

OSR-060 *(depends on other components in an M2M system where XMPP is being used),*
OSR-061 *(depends on other components in an M2M system where XMPP is being used),*

6.6 WebSocket Protocol

The following clauses describe the WebSocket Protocol.

6.6.1 Background

The HTTP protocol is widely used for Web Services in different ways. One of those usages is standardized as WebSocket Protocol.

WebSocket Protocol provides an alternative way to 'HTTP polling' for two-way communication, which transmits data with upgrading established HTTP connection with remote host.

WebSocket Protocol was originally introduced as part of HTML5 specification, but later the transport protocol specification was standardized as the extension for HTTP in IETF.

6.6.2 Status

WebSocket Protocol specification is published as RFC6455 [i.40] (PROPOSED STANDARD) by the IETF in 2011.

SIP over WebSocket and XMPP over WebSocket are under development.

6.6.4 Intended use

WebSocket Protocol provides relatively simple transport for two-way communication with remote host, which can coexist with HTTP and deployed HTTP infrastructure.

Unlike HTTP 2.0, which also possible to two-way communication over HTTP connection, WebSocket is designed for use in Ajax applications.

6.6.5 Deployment Trend

Following are some of implementations which include WebSocket support:

6.6.5.1 Server-Side Implementations

- Apache Tomcat 7 (or later)
- Jetty 8 (or later)
- Tornado
- em-websocket (Ruby)
- gevent-websocket (Python)
- Node.js

6.6.5.2 Client-Side Implementations

- Google Chrome 14.0 (or later)
- Firefox 10.0 (or later)
- Safari 6.0 (or later)
- Internet Explorer 10.0 (or later)

- Cordova (AKA PhoneGap)

6.6.6 Key features

NOTE: The following text is excerpted from Section “1.5 Design Philosophy” of RFC6455 [i.40]

The WebSocket Protocol is designed on the principle that there should be minimal framing (the only framing that exists is to make the protocol frame-based instead of stream-based and to support a distinction between Unicode text and binary frames). It is expected that metadata would be layered on top of WebSocket by the application layer, in the same way that metadata is layered on top of TCP by the application layer (e.g., HTTP).

Conceptually, WebSocket is really just a layer on top of TCP that does the following:

- adds a web origin-based security model for browsers
- adds an addressing and protocol naming mechanism to support multiple services on one port and multiple host names on one IP address
- layers a framing mechanism on top of TCP to get back to the IP packet mechanism that TCP is built on, but without length limits
- includes an additional closing handshake in-band that is designed to work in the presence of proxies and other intermediaries

Other than that, WebSocket adds nothing. Basically it is intended to be as close to just exposing raw TCP to script as possible given the constraints of the Web. It's also designed in such a way that its servers can share a port with HTTP servers, by having its handshake be a valid HTTP Upgrade request. One could conceptually use other protocols to establish client-server messaging, but the intent of WebSockets is to provide a relatively simple protocol that can coexist with HTTP and deployed HTTP infrastructure (such as proxies) and that is as close to TCP as is safe for use with such infrastructure given security considerations, with targeted additions to simplify usage and keep simple things simple (such as the addition of message semantics).

The protocol is intended to be extensible; future versions will likely introduce additional concepts such as multiplexing.

6.6.9 Security

Security feature on WebSocket can be independent, but HTTP can provide necessary protections in the transport layer.

6.6.10 Dependencies

- WebSocket Protocol depends on HTTP (1.1 or later) protocol.
- TCP stack should be provided for HTTP communication.

6.6.11 Benefits and Constraints

6.6.11.1 Benefits

- Allowing co-existence with existing Web infrastructures
- Reuse security solutions which is applied on HTTP Servers
- HTML5 browser can supports WebSocket natively (seamless integration on HTTP based application)

6.6.11.2 Constraints

- Leaving WebSocket connection opened for long time periods may cause DoS attack risk
- Server-side WebSocket implementation can host limited number of connections

6.7 Bluetooth® Wireless Technology

This clause outline the background and practical use and characteristics of Bluetooth® with particular focus on the Bluetooth Low Energy (BLE) version that is well-suited to M2M type of applications. There is also a description of the application layer and attribute protocol based on the GATT (Generic Attribute Profile) interactions and data structure (Services and Characteristics), that is included in the Bluetooth Smart version 4.0. Finally the GATT although a general purpose solution and well specified to be interoperable is not immediately available of plain HTTP. For this reason there is a host of developer resources provided by the Bluetooth SIG developer portal that outline how to develop for Bluetooth Low Energy and GATT for various popular OS and platforms. A more recent activity (Whitepaper, publication in process) also present a practical method to transfer and interact with GATT devices using HTTP and JSON data over Internet and Web.

6.7.1 Background

Bluetooth® technology is a wireless communications system intended to replace the cables connecting electronic devices. Bluetooth technology is powerful, low energy and lower in cost to make your development faster and easier.

The Bluetooth Core System consists of an RF transceiver, baseband, and protocol stack. The system offers services enabling the connection of devices and the exchange of a variety of classes of data between these devices. Many features of the core specification are optional, allowing product differentiation.

The most recent enhancement, Bluetooth v4.0, is like three specifications in one—Classic Bluetooth technology, Bluetooth low energy technology, and Bluetooth high speed technology—all of which can be combined or used separately in different devices according to their functionality.

To use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles. Bluetooth profiles are definitions of possible applications and specify general behaviours that Bluetooth enabled devices use to communicate with other Bluetooth devices. There is a wide range of Bluetooth profiles describing many different types of applications or use cases for devices. By following the guidance provided by the Bluetooth specification, developers can create applications to work with other Bluetooth devices.

6.7.2 Status

When the Bluetooth® SIG announced the formal adoption of Bluetooth Core Specification version 4.0, it included the Bluetooth Smart (low energy) feature. This final step in the adoption process opened the door for qualification of all Bluetooth product types to version 4.0.

Bluetooth v.1.2 is endorsed and ratified as IEEE 802.15.1; this is based on the standard issued in 2005). Recent versions of Bluetooth now known as 4.0 – or “Bluetooth Smart” are openly & publically available thought the Bluetooth SIG. The specifications may be endorsed ratified & published. Publication is an option, if preferred by an SDO or partnership project. The Bluetooth SIG is encourages direct liaison with oneM2M.

The Bluetooth SIG considers that Version 4.0 Bluetooth Smart is frozen and stable. It is being tested against devices for conformance, and new profiles are being added. However the core specification is being used and updated toward version 4.1

6.7.2.1 Bluetooth Core Specification 4.1

Bluetooth® Core Specification 4.1 [i.91] is an important evolutionary update to the Bluetooth Core Specification. It rolls up adopted Bluetooth Core Specification Addenda (CSA 1, 2, 3 & 4) while adding new features and benefits as well. Bluetooth 4.1 improves usability for consumers, empowers innovation for product developers and extends the technology's foundation as an essential link for the Internet of Things.

6.7.2.2 Improving Usability - Bluetooth 4.1

Bluetooth 4.1 extends the Bluetooth brand promise to provide consumers with a simple experience that "just works." Major usability updates come in three areas:

- Coexistence — engineered to work seamlessly and cooperatively with the latest generation cellular technologies like LTE. Bluetooth and LTE radios can communicate in order to ensure transmissions are coordinated and therefore reduce the possibility of near-band interference. The coordination between the two technologies happens automatically, while the consumer experiences the high quality they expect.

- Better Connections — provides manufacturers with more control over creating and maintaining Bluetooth connections by making the reconnection time interval flexible and variable. This improves the consumer experience by allowing devices to reconnect automatically when they are in proximity of one another. The consumer can leave the room and upon returning, two recently used devices reconnect without user intervention.

- Improved Data Transfer — Bluetooth Smart technology provides bulk data transfer. For example, through this new capability, sensors, which gathered data during a run, bike ride or swim, transfer that data more efficiently when the consumer returns home.

The latest Bluetooth 4.1 technical details, tools and other information including a brand guide, visit: https://www.bluetooth.org/en-us/specification/adopted-specifications.

6.7.3 Category and Architectural Style

The Bluetooth® core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as one common service layer protocol, the service discovery protocol (SDP) and the overall profile requirements specified in the generic access profile (GAP). A complete Bluetooth application requires a number of additional services and higher layer protocols defined in the Bluetooth specification.

To use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles. Bluetooth profiles are definitions of possible applications and specify general behaviours that Bluetooth enabled devices use to communicate with other Bluetooth devices. There is a wide range of Bluetooth profiles describing many different types of applications or use cases for devices. By following the guidance provided by the Bluetooth specification, developers can create applications to work with other Bluetooth devices.

At a minimum, each Bluetooth profile contains information on the following topics:

- Dependencies on other profiles

- Suggested user interface formats

- Specific parts of the Bluetooth protocol stack used by the profile. To perform its task, each profile uses particular options and parameters at each layer of the stack and this may include, if appropriate, an outline of the required service record

6.7.4 Intended use - Personal Area Network protocols

Bluetooth Smart (low energy) technology allows enhancement of devices like watches, toothbrushes or toys with Bluetooth wireless technology. It also provides the ability for developers to incorporate new functionalities into devices already enabled by Bluetooth technology such as sports & fitness, health care, human interface (HIDs) and entertainment devices. For example, sensors in pedometers and glucose monitors will only run low energy technology. These single mode devices benefit from the power savings provided by v4.0 as well as the low cost implementation. Watches take advantage of both low energy technology while collecting data from body-worn fitness sensors and Classic Bluetooth technology when sending that information to a PC, or displaying caller ID information when wirelessly connected to a smartphone. Smartphones and PCs, which support the widest range of use cases for the specification, utilizing the full dual-mode package with Classic, low energy and high speed versions of the technology running side by side.

6.7.5 Deployment Trend - Bluetooth and Bluetooth Smart (low energy)

Originally intended to be a wireless replacement for cables on phones, headsets, keyboards and mice, Bluetooth technology now goes way beyond that. Bluetooth technology is bringing everyday devices into a digital and connected world. In the health and fitness market, the use cases vary widely — from sensors that monitor activity levels to medical and wellness devices that monitor healthcare, like a glucometer, inhaler or toothbrush. The top-selling Smartphones, PCs and tablets all support Bluetooth technology. In-vehicle systems give the ability to make phone calls, send texts, and even make dinner reservations. The Bluetooth SIG is also seeing developments where drivers will monitor important information like vehicle diagnostics, traffic, even driver health — all in real time. Bluetooth technology is creating opportunities for companies to develop solutions that make a consumer's life better.

Bluetooth Smart and Bluetooth Smart Ready are extensions of the original Bluetooth brand introduced in 2011. The Smart and Smart Ready designations indicate compatibility of products using the low energy feature of the Bluetooth v4.0 specification. A Bluetooth Smart Ready product connects to both classic Bluetooth and Bluetooth Smart low energy products. By contrast, a Bluetooth Smart product collects data and runs for months or years on a tiny battery. Think of a Smart product as a sensor that works for a long time without changing the battery (like a fitness heart rate monitor) and a Smart Ready product as a collector (like a smart phone or tablet receiving the information and displaying it in an application).

6.7.5.1 Bluetooth Smart (low energy) Technology

When the Bluetooth® SIG announced the formal adoption of Bluetooth Core Specification version 4.0, it included the hallmark Bluetooth Smart (low energy) feature. This final step in the adoption process opened the door for qualification of all Bluetooth product types to version 4.0.

Bluetooth Smart (low energy) technology allows it to be included in devices like watches, toothbrushes or toys to enable the connectivity using Bluetooth wireless technology. It also provides the ability for developers to incorporate new functionalities into devices already enabled by Bluetooth technology such as sports & fitness, health care, human interface (HIDs) and entertainment devices. For example, sensors in pedometers and glucose monitors will only run low energy technology. These single mode devices benefit from the power savings provided by v4.0 as well as the low cost implementation. Watches take advantage of both low energy technology while collecting data from body-worn fitness sensors and Classic Bluetooth technology when sending that information to a PC, or displaying caller ID information when wirelessly connected to a smartphone. Smartphones and PCs, which support the widest range of use cases for the specification, utilizing the full dual-mode package with Classic, low energy and high speed versions of the technology running side by side.

6.7.5.2 Bluetooth High Speed Wireless Technology

Bluetooth high speed wireless technology delivers new opportunities in the home entertainment and consumer electronics markets. By enabling wireless users to quickly send video, music and other large files between devices, Bluetooth high speed wireless technology provides a richer experience while maintaining the same familiar user interface.

Key features of Bluetooth high speed wireless technology include:

- Power Optimization. The new Bluetooth technology reduces power consumption. The high speed radio is used only when necessary, which means longer battery life for your devices
- Improved Security. The Generic Alternate MAC/PHY in Bluetooth high speed enables the radio to discover other high speed devices only when they are needed in the transfer of music, video and other large data files. This decreases power consumption and increases radio security
- Enhanced Power Control. Drop-out reduction is now a reality: enhanced Bluetooth technology makes power control faster and reduces the impact of a power or signal loss. Users are now less likely to experience a dropped headset connection – even when a phone is deep inside a coat pocket or tote
- Lower Latency Rates. Unicast Connectionless Data (UCD) improves the user’s speed experience by moving small amounts of data faster, which lowers latency rates

6.7.5.3 Enabling the Internet of Things

By adding a standard means to create a dedicated channel, which could be used for IPv6 communications in the Core Specification, the groundwork is laid for future protocols providing IP connectivity. With the rapid market adoption of Bluetooth Smart and the coming addition of IP connectivity, all signs point to Bluetooth as a fundamental wireless link in the Internet of Things. These updates make it possible for Bluetooth Smart sensors to also use IPv6, giving developers and OEMs the flexibility they need to ensure connectivity and compatibility

6.7.6 Key features

- Bluetooth wireless technology is geared towards voice and data applications
- Bluetooth wireless technology operates in the unlicensed 2.4 GHz spectrum

- The range of Bluetooth wireless technology is application specific. The Bluetooth Specification mandates operation over a minimum distance of 10 meters or 100 meters depending on the Bluetooth device class, but there is not a range limit for the technology. Manufacturers may tune their implementations to support the distance required by the use case they are enabling.
- The peak data rate with EDR is 3 Mbps

NOTE: The term Enhanced Data Rate (EDR) is used to describe π/4-DPSK and 8DPSK schemes, each giving 2 and 3 Mbit/s respectively

- EDR Profiles in steps 10 to 100 meters.
- Bluetooth wireless technology is able to penetrate solid objects
- Bluetooth technology is omni-directional and does not require line-of-sight positioning of connected devices
- Security has always been and continues to be a priority in the development of the Bluetooth specification. The Bluetooth specification allows for three modes of security, see below for security.
- Bluetooth 4.1 provides a Developer Toolkit.
- A single device may act as both a Bluetooth Smart peripheral and a Bluetooth Smart Ready hub at the same time.
- Coexistence — engineered to work seamlessly and cooperatively with the latest generation cellular technologies like LTE. Bluetooth and LTE radios may communicate in order to ensure transmissions are coordinated and therefore reduce the possibility of near-band interference.
- Better Connections — provides manufacturers with more control over creating and maintaining Bluetooth connections by making the reconnection time interval flexible and variable.
- Improved Data Transfer — Bluetooth Smart technology provides bulk data transfer.
- Adding a standard means to create a dedicated channel the adoption of Bluetooth Smart and the IP connectivity makes Bluetooth a fundamental wireless link in the Internet of Things.

Bluetooth Smart (low energy) wireless technology features:

- Ultra-low peak, average and idle mode power very low consumption
- Ability to run for years on standard coin-cell batteries
- Multi-vendor interoperability
- Enhanced range (EDR Profiles in steps 10 to 100+ meters).

Technology	Bluetooth BR/EDR/HS Technology	Bluetooth Low Energy Technology
Radio Frequency	2.4 GHz ISM	2.4 GHz ISM
Range	10 to 100 meters	10 to 100+ meters
Data Rate	1-3 Mbps (Classic) >400 Mbps (AMP, 802.11n)	1 Mbps
Nodes/Active Slaves	7 / 16777184	Unlimited
Security	56b E0 (classic)/128b AES (AMP) and applications layer user defined	128b AES and application layer user defined
Robustness	Adaptive frequency hopping,	FEC Adaptive frequency hopping
Latency (from non-connected state)	100ms	<3ms

Regulatory Acceptance	Worldwide	Worldwide
Voice Capable	Yes	No
Network Topology	Scatternet	Star-bus
Power Consumption	1 as the reference, x10 for AMP	0.01 to 0.5 (use case dependent)
Service Discovery	Yes	Yes

Table 6.7.1: Table of Bluetooth capabilities

6.7.7 Protocol Stack

The Bluetooth 4.0 specification uses a service-based architecture based on the attribute protocol (ATT). All communication in low energy takes place over the Generic Attribute Profile (GATT). An application or another profile uses the GATT profile so a client and server can interact in a structured way.

The server contains a number of attributes, and the GATT Profile defines how to use the Attribute Protocol to discover, read, write and obtain indications. These features support a service-based architecture. The services are used as defined in the profile specifications. GATT enables to expose service and characteristics defined in the profile specification.

The GATT profile is also part of the core and defined in the core specification.

Profiles: The first specification of Bluetooth 4.0 low energy wireless technology included profiles to optimize its functionality for a specific group of products.

Adopted GATT based Bluetooth Profiles and Services: The GATT architecture makes it easy to both create and implement new profiles. Many new profiles are under development so this continues to grow. The simplicity of implementing the profiles contributes to a rapid growth of applications and embedded devices supporting these.

Generic Attribute Profile (GATT) is built on top of the Attribute Protocol (ATT) and establishes common operations and a framework for the data transported and stored by the Attribute Protocol. GATT defines two roles: Server and Client. The GATT roles are not necessarily tied to specific GAP roles and but may be specified by higher layer profiles. GATT and ATT are not transport specific and can be used in both BR/EDR and LE. However, GATT and ATT are mandatory to implement in LE since it is used for discovering services

The GATT server stores the data transported over the Attribute Protocol and accepts Attribute Protocol requests, commands and confirmations from the GATT client. The GATT server sends responses to requests and when configured, sends indication and notifications asynchronously to the GATT client when specified events occur on the GATT server. GATT also specifies the format of data contained on the GATT server.

Attributes, as transported by the Attribute Protocol, are formatted as services and characteristics. Services may contain a collection of characteristics. Characteristics contain a single value and any number of descriptors describing the characteristic value.

With the defined structure of services, characteristics and characteristic descriptors a GATT client that is not specific to a profile can still traverse the GATT server and display characteristic values to the user. The characteristic descriptors can be used to display descriptions of the characteristic values that may make the value understandable by the user.

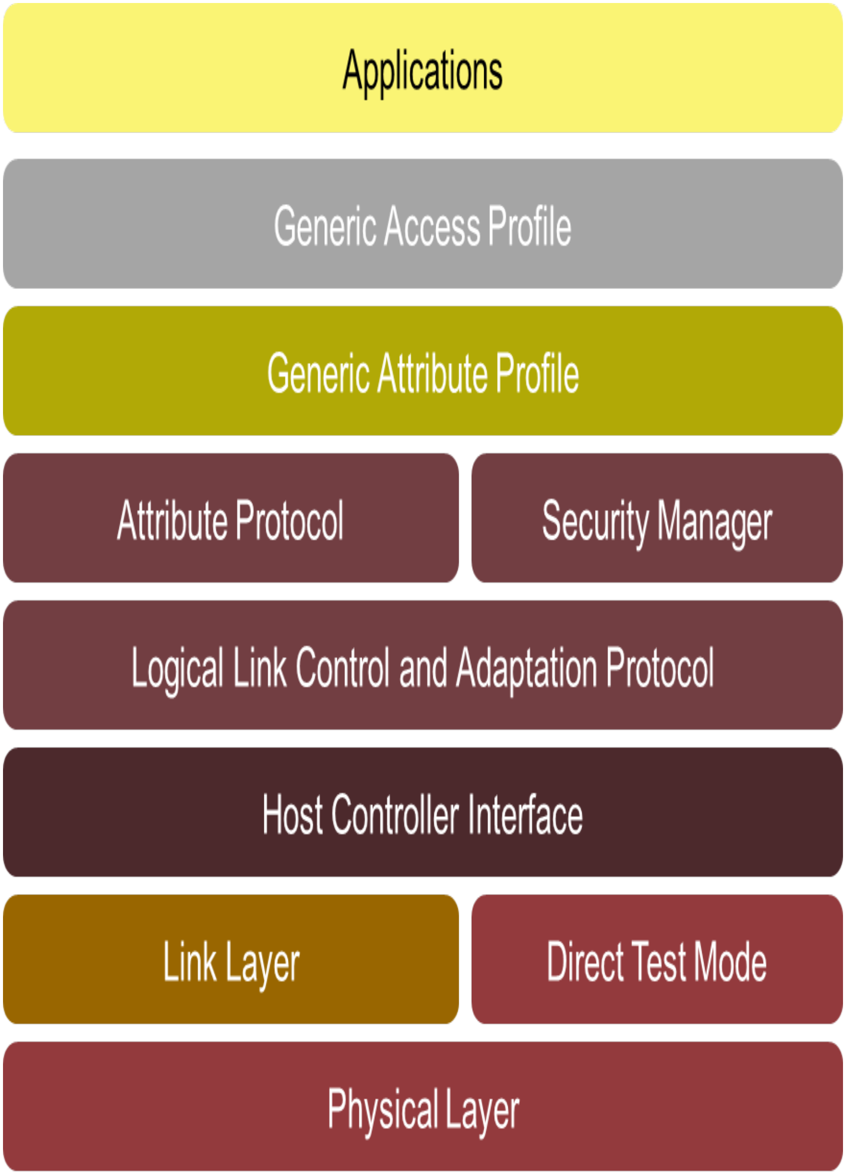


Figure 6.7: Bluetooth Protocol Architecture

Below are links to the specifications for the current list of profiles supported by GATT in the above protocol architecture.

GATT-Based Specifications (Qualifiable)		**Adopted Versions**
ANP	Alert Notification Profile	1.0
ANS	Alert Notification Service	1.0
BAS	Battery Service	1.0
BLP	Blood Pressure Profile	1.0
BLS	Blood Pressure Service	1.0
CPP	Cycling Power Profile	1.0
CPS	Cycling Power Service	1.0
CSCP	Cycling Speed and Cadence Profile	1.0

CSCS	Cycling Speed and Cadence Service	1.0
CTS	Current Time Service	1.0
DIS	Device Information Service	1.0
FMP	Find Me Profile	1.0
GLP	Glucose Profile	1.0□
GLS	Glucose Service	1.0□
HIDS	HID Service	1.0□
HOGP	HID over GATT Profile	1.0□
HTP	Health Thermometer Profile	1.0
HTS	Health Thermometer Service	1.0
HRP	Heart Rate Profile	1.0
HRS	Heart Rate Service	1.0
IAS	Immediate Alert Service	1.0
LLS	Link Loss Service	1.0
LNP	Location and Navigation Profile	1.0□
LNS	Location and Navigation Service	1.0□
NDCS	Next DST Change Service	1.0
PASP	Phone Alert Status Profile	1.0
PASS	Phone Alert Status Service	1.0
PXP	Proximity Profile	1.0
RSCP	Running Speed and Cadence Profile	1.0□
RSCS	Running Speed and Cadence Service	1.0□
RTUS	Reference Time Update Service	1.0
ScPP	Scan Parameters Profile	1.0□
ScPS	Scan Parameters Service	1.0□
TIP	Time Profile	1.0
TPS	Tx Power Service	1.0

Table 6.7.2: List of Profiles

The link layer provides low power idle mode operation, simple device discovery and reliable point-to-multipoint data transfer with advanced power-save and encryption functionalities.

The following table is a list of the Basic rate/Enhanced Data rate technology profiles adopted in Bluetooth version 4.1 that may be found at https://www.bluetooth.org/en-us/specification/adopted-specifications.

•

BR/EDR Profiles		**Description**
A2DP	Advanced Audio Distribution Profile	describes how stereo quality audio can be streamed from a media source to a sink.

AVRCP	Audio/Video Remote Control Profile	is designed to provide a standard interface to control TVs, stereo audio equipment, or other A/V devices. This profile allows a single remote control (or other device) to control all A/V equipment to which a user has access.
BIP	Basic Imaging Profile	defines how an imaging device can be remotely controlled, how an imaging device may print, and how an imaging device can transfer images to a storage device.
BPP	Basic Printing Profile	allows devices to send text, e-mails, v-cards, images or other information to printers based on print jobs.
DI	Device ID Profile	provides additional information above and beyond the *Bluetooth* Class of Device and to incorporate the information into both the Service Discovery Profile (SDP) record and the EIR response.
DUN	Dial-Up Network Profile	provides a standard to access the Internet and other dial-up services via *Bluetooth* technology.
FTP	File Transfer Profile	defines how folders and files on a server device can be browsed by a client device.
GAVDP	Generic Audio/Video Distribution Profile	provides the basis for A2DP and VDP, which are the basis of the systems designed for distributing video and audio streams using *Bluetooth* technology.
GOEP	Generic Object Profile	is used to transfer an object from one device to another.
HFP	Hands-Free Profile	HFP describes how a gateway device can be used to place and receive calls for a hand-free device.
HCRP	Hard Copy Cable Replacement Profile	defines how driver-based printing is accomplished over a *Bluetooth* wireless link.
HDP	Health Device Profile	enables Healthcare and Fitness device usage models.
HSP	Headset Profile	describes how a *Bluetooth* enabled headset should communicate with a *Bluetooth* enabled device.
HID	Human Interface Device Profile	defines the protocols, procedures and features to be used by *Bluetooth* keyboards, mice, pointing and gaming devices and remote monitoring devices.
MAP	Message Access Profile	defines a set of features and procedures to exchange messages between devices.
MPS	Multi Profile	defines a set of features and procedures between Multiple Profiles Single Device and Multiple Profiles Multiple Devices
OPP	Object Push Profile	defines the roles of push server and push client.
PBAP	Phone Book Access Profile	defines the procedures and protocols to exchange Phone Book objects between devices.
PAN	Personal Area Networking Profile	describes how two or more *Bluetooth* enabled devices can form an ad-hoc network and how the same mechanism can be used to access a remote network through a network access point.
SAP	SIM Access Profile	defines the protocols and procedures that shall be used to access a GSM SIM card, a UICC card or an R-UIM card via a Bluetooth link.
SDAP	Service Discovery Application Profile	describes how an application should use SDP to discover services on a remote device.
SPP	Service Port Profile	defines how to set-up virtual serial ports and connect two Bluetooth enabled devices.
SYNC	Synchronization Profile	used in conjunction with GOEP to enable synchronization of calendar and address information (personal information manager (PIM) items) between Bluetooth enabled devices.

VDP	Video Distribution Profile	defines how a Bluetooth enabled device streams video over Bluetooth wireless technology.

Table 6.7.3: List of BR/EDR Profiles

The following table is a list of the Core Bluetooth specification version 4.1 that may be found at https://www.bluetooth.org/en-us/specification/adopted-specifications.

Specification	Adopted Date	Notes
Core Specification Addendum (CSA) 4	12 February 2013	Refer to the Mixing of Specification Versions Part for applicability
Core Specification Supplement (CSS) v3	12 February 2013	
Core Specification Addendum (CSA) 3	24 July 2012	Refer to the Mixing of Specification Versions Part for applicability
Core Specification Addendum (CSA) 2	27 December 2011	Refer to the Mixing of Specification Versions Part for applicability
Core Specification Supplement (CSS) v4	03 December 2013	
Core Version 4.1	03 December 2013	

Table 6.7.4: Bluetooth Core Specifications

For the latest Bluetooth 4.1 technical details, tools, visit: https://www.bluetooth.org/en-us/specification/adopted-specifications.

6.7.7.1 Bluetooth Smart (low energy) Single mode and dual mode

Bluetooth Smart (low energy) wireless technology contains two equally important implementation alternatives: single mode and dual mode:

- Small devices like watches and sports sensors use the single mode Bluetooth Smart (low energy) implementation.
- Dual mode implementations use parts of the Bluetooth hardware, sharing one physical radio and antenna.

6.7.8 Data Model

The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile. The top level of the hierarchy is a profile. A profile is composed of one or more services necessary to fulfil a use case. A service is composed of characteristics or references to other services. Each characteristic contains a value and may contain optional information about the value. The service and characteristic and the components of the characteristic (i.e., value and descriptors) contain the profile data and are all stored in attributes on the server.

A service is a collection of data and associated behaviours to accomplish a particular function or feature of a device or portions of a device. A service may reference other primary or secondary services and/or a set of characteristics that make up the service.

There are two types of services: primary and secondary. A primary service provides the primary functionality of a device. A secondary service provides auxiliary functionality of a device and is referenced from at least one primary service on the device.

To maintain backward compatibility with earlier clients, later revisions of a service definition can only add new referenced services or optional characteristics. Later revisions of a service definition are also forbidden from changing behaviours from previous revision of the service definition. Services may be used in one or more profiles to fulfil a particular use case.

A referenced service is a method incorporating another service definition on the server as part of the service referencing it. When a service references another service, the entire referenced service becomes part of the new service including any nested referenced services and characteristics. The referenced service still exists as an independent service. There are no limits to the depth of nested references.

A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. A characteristic definition contains a characteristic declaration, characteristic properties, and a value. It may also contain descriptors that describe the value or permit configuration of the server with respect to the characteristic value.

6.7.9 Security

Bluetooth® Smart (low energy) technology has some security differences with respect to BR/EDR security features such as Secure Simple Pairing. The association models are similar to Secure Simple Pairing from the user perspective and have the same names but differences in the quality of the protection provided.

The overall goal of keeping the cost of the controller and the complexity of a slave device to a minimum was used in making compromises on security capabilities in Bluetooth Smart (low energy) technology.

Bluetooth Smart (low energy) technology uses three association models referred to as Just Works, Out of Band and Passkey Entry. Bluetooth low energy technology does not have an equivalent of Numeric Comparison. Each of these association models is similar to Secure Simple Pairing with the following exception; Just Works and Passkey Entry do not provide any passive eavesdropping protection. This is because Secure Simple Pairing uses Elliptic Curve Diffie-Hellman and Bluetooth Smart (low energy) does not. The use of each association model is based on the I/O capabilities of the devices in a similar manner as Secure Simple Pairing.

Bluetooth Smart (low energy) technology supports a feature that reduces the ability to track a Bluetooth device over a period of time by changing the address on a frequent basis. The privacy feature is not used in the GAP discovery mode and procedures but it is used when supported during connection mode and connection procedures.

6.7.10 Dependencies

Support for IPv4 & IPv6, is provided in Bluetooth smart version 4.0 & 4.1, as it is globally accepted with worldwide regulatory approval the radio availability & lack of interference make it an ideal technology, as a short range radio access network to IP. The updates in version 4.1 & beyond make it possible for Bluetooth Smart sensors to also use IPv6, giving developers and OEMs the flexibility they need to ensure connectivity and compatibility.

6.7.11 Benefits and Constraints

Bluetooth Smart (low energy) wireless technology contains two equally important implementation alternatives: single mode and dual mode:

- Small devices like watches and sports sensors use the single mode Bluetooth Smart (low energy) implementation.
- Dual mode implementations use parts of the Bluetooth hardware, sharing one physical radio and antenna.

Bluetooth high speed wireless technology delivers new opportunities in the home entertainment and consumer electronics markets. By enabling wireless users to quickly send video, music and other large files between devices, Bluetooth high speed wireless technology provides a richer experience while maintaining the same familiar user interface.

6.7.11.1 Benefits

Bluetooth 4.1 extends the Bluetooth Smart development environment by providing product and application developers with even more flexibility to enable the creation of products that can take on multiple roles. With this new capability, a single device acts as both a Bluetooth Smart peripheral and a Bluetooth Smart Ready hub at the same time. For example, a smart watch acts as a hub gathering information from a Bluetooth Smart heart rate monitor while simultaneously acting as a peripheral to a smartphone — displaying new message notifications from the phone.

Bluetooth 4.1 delivers this type of flexibility to Bluetooth Smart devices and application developers.

By adding a standard means to create a dedicated channel, which could be used for IPv6 communications in the Core Specification, the groundwork is laid for future protocols providing IP connectivity. With the rapid market adoption of

Bluetooth Smart and the coming addition of IP connectivity, all signs point to Bluetooth as a fundamental wireless link in the Internet of Things.

- Coexistence — engineered to work seamlessly and cooperatively with the latest generation cellular technologies like LTE. Bluetooth and LTE radios may communicate in order to ensure transmissions are coordinated and therefore reduce the possibility of near-band interference.
- Better Connections — provides manufacturers with more control over creating and maintaining Bluetooth connections by making the reconnection time interval flexible and variable.
- Improved Data Transfer — Bluetooth Smart technology provides bulk data transfer.
- Adding a standard means to create a dedicated channel the adoption of Bluetooth Smart and the IP connectivity makes Bluetooth a fundamental wireless link in the Internet of Things.

6.7.11.2 Constraints

The capabilities & constraints are summarized in Table 6.7.1 (above): Table of Bluetooth capabilities, above. Unicast between paired devices, Multicast not supported to groups of paired devices. Not intended to replace access & core network protocols.

6.7.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by Bluetooth is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and Bluetooth comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.7.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-004, OSR-008, OSR-009, OSR-012, OSR-014, OSR-016, OSR-019, OSR-024, OSR-25, OSR-26, OSR-028, OSR-029, OSR-030, OSR-031, OSR-034, OSR-035, OSR-037, OSR-040, OSR-041, OSR-047, OSR-050, OSR-058, OSR-060, OSR-061, OSR-062, OSR-070, SER-002, SER-003, SER-008, SER-009

6.7.12.2 Partially Supported Requirements

OSR-005, OSR-006, OSR-007, OSR-010, OSR-011, OSR-013, OSR-015, OSR-017, OSR-020, OSR-032, OSR-033, OSR-038, OSR-039, OSR-042, SER-004, SER-005

6.7.12.3 Unsupported Requirements

OSR-018 (this is for cellular devices) not in scope of Bluetooth,
OSR-022, OSR-044, OSR-045, & OSR-044 Bluetooth is a capilary access,

SER-004 to SER-006
(*These requirements are for UICC based devices. Some enhancements are needed for support of UICC based devices with Bluetooth, these are proposed in the coming Bluetooth release.)*

6.8 Data Distribution Service (DDS) for Real-Time Systems

Data Distribution Service (DDS) for Real-Time Systems is a peer-to-peer publish/subscribe communications service for real-time and non-real-time data. DDS is comprised of a standardized application API known as Data Centric Publish Subscribe (DCPS) and a standardized wire protocol called Real Time Publish Subscribe (RTPS). The standardized interfaces provide for both vendor and platform independent middleware implementations. In order to be more concise we shall be using the term DDS all of these protocols and interfaces and be specific about DCPS, RTPS when appropriate.

6.8.1 Background

DDS is a data oriented middleware architecture standardized and managed by the Object Management Group (OMG).

- Data Distribution Service for Real-time Systems (DDS) Version 1.2, adopted January 2007 [i.41]
- The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification (DDSI), Version 2.1, adopted June 2008 [i.42]

In addition there are several activities for expanding the extensibility, security and development support.

6.8.1.1 Extensibility, Security and Development support

There are several activities for expanding the extensibility, security and development support:

- Extensible and Dynamic Topic Types for DDS (DDS X-Types).
  This standardizes how DDS manages data-types and how an application designer can define and use those types. Specifically the standard covers:
    - Type System: Specifies a model for the types that can be legally defined and associated with Topics.
    - Language Binding: Specifies the alternative programming-language mechanisms an application can use to construct and introspect types as well as objects of those types. These mechanisms include in part a Dynamic API that allows an application to interact with types and data without compile-time knowledge of the type.
    - Type Representation: Specifies the ways in which a type definition can be externalized so that it may be stored in a file or communicated over the network.
    - Data Representation: Specifies the ways in which a data sample of a given type can be externalized so that it can be stored in a file or communicated over the network. This is also commonly referred as “data serialization” or “data marshalling”.
- Web-Enabled DDS (WEB-DDS)
  Standardises how web client applications can participate in the DDS global data space in a way that is portable across implementations. The specification includes (1) a platform-independent Abstract Interaction Model of how web- clients should access a DDS System and (2) a set of mappings to specific web platforms that realize the PIM in terms of standard web technologies and protocols. These mappings include a RESTful mapping of DDS entities to HTTP REST resources. This specification also defines an XML document format to represent DDS Domains and DDS Entities.
- RPC over DDS, Currently under RFP process, revised submission May 20, 2013.
  Specification to provide request/response communications pattern for remote procedure calls. Provide auto-generation of client/server code, request/reply topic, and development environment. Similar approach to Apache Thrift but easier to use and configurable QOS.
- DDS Security, Currently under RFP process, revised submission June 13, 2013.
  Specification defines the Security Model and Service Plugin Interface (SPI) architecture for compliant DDS implementations. The DDS Security Model is enforced by the invocation of these SPIs by the DDS implementation

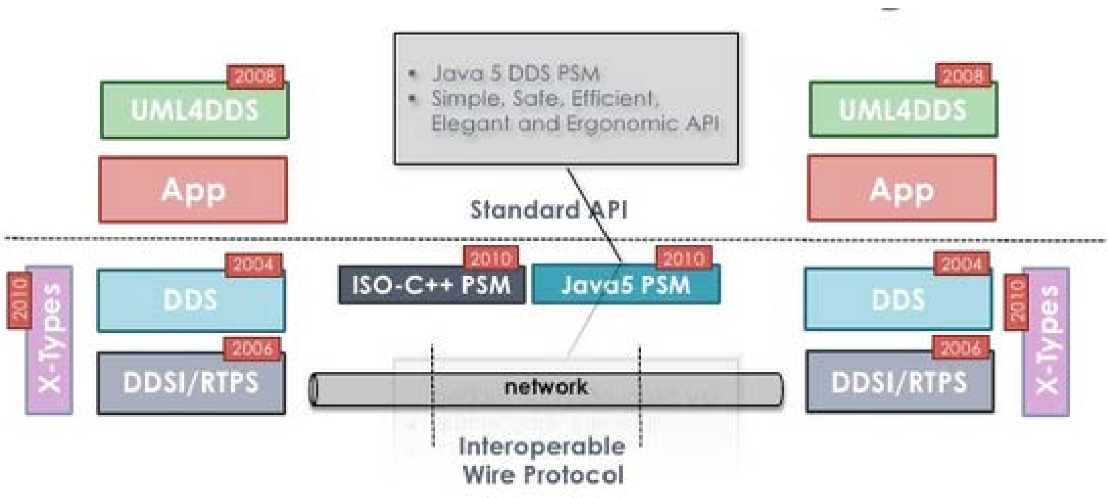


Fig 6.8.1: Present and Future of DDS

ref: http://www.slideshare.net/Angelo.Corsaro/the-present-and-future-of-dds

There are currently more than 9 implementations of the DDS standard, with participants from PrismTech, RTI, TwinOaks Computing and Gallium Visual Systems.

6.8.2 Status

Data Distribution Service for Real-Time Systems was adopted in June 2003 and finalized in June 2004. Revisions followed in 2005, 2006 with the final 1.2 specification delivered June 2007. DDSI-RTPS was adopted in July 2006, revised in 2007 for release 2.1. The OMG is currently working on several proposals related to security and RPC.

6.8.3 Category and Architectural Style

DDS is specific to a "Data Centric" style of distributed communications, which leverages a publish/subscribe communications model to connect anonymous information producers with information consumers. The overall distributed application is composed of processes called "participants," each running in a separate global data space, possibly on different computers. DDS handles all transport functions including: addressing, data serialization, reliable delivery, flow control in a platform independent way.

DDS defines a bi-directional communications relationship between publishers and subscribers. The unit of information exchanged between publishers and subscribers is called a Topic. The communications are decoupled in space (nodes can be anywhere), time (nodes and topics can start, join, or leave in any order at any time), and flow (delivery may be "best effort", reliable, or at controlled bandwidth).

DDS provides an expressive service level specification via QoS (Quality of Service) that can act upon participants, properties and topics.

To increase scalability, topics may contain multiple independent data channels identified by "keys." This allows nodes to subscribe to many, possibly thousands, of similar data streams with a single subscription. When the data arrives, the middleware can sort it by the key and deliver it for efficient processing.

DDS also provides a "state propagation" model. This model allows nodes to treat DDS-provided data structures like distributed shared-memory objects, with local caches efficiently updated only when the underlying data changes. There are facilities to ensure coherent and ordered state updates.

DDS is fundamentally designed to work over unreliable transports, such as UDP or wireless networks. No facilities require central servers or special nodes. Efficient, direct, peer-to-peer communications, or even multicasting, can implement every part of the model.

6.8.4 Intended use

The following details the various types of use cases that are addressed by the use of DDS in real-time M2M systems.

DDS is highly suitable for direct Device2Device (D2D) communications when there is a need for distributing data to a large fan-out of consumers with real-time requirements. DDS allows for implementers to choose the most appropriate trade-offs to meet their application objectives through configurable reliability, multicast support, transport priority and pervasive redundancy. By leveraging the finely controlled QoS, designers can provide integration of complex systems with modules that require differing update rates, reliability, and bandwidth control. These controls are available on a per- node, or per-stream basis.

With the standardization of RPC over DDS, alternative pattern to data distribution such as invocation can also be beneficial to M2M.

Because of the Global Data Space, DDS systems can easily share information to tie together complex applications.

For example, a ship control system that includes high-bandwidth data sources and sinks, slowly updated graphical user interfaces (GUIs), and long-term logging facilities will make good use of the DDS QoS parameters. While the control system is updated at high rates, the GUI can subscribe at a reduced rate to display the latest state of the system. The logger receives every update reliably, although perhaps with greater latency, to save a complete record of the system operation.

6.8.5 Deployment Trend

DDS can be found in applications ranging from large navy ships to single embedded sensor applications. It is deployed in healthcare devices, unmanned systems such as UAV's, financial banking applications, asset tracking solutions, remote diagnostic monitoring for power substations and many other types of applications. The typical type of application using DDS today is one where low latency, highly deterministic communications is desired between devices.

6.8.6 Key features

6.8.6.1 Platform and Language Independence

DDS decouples the platform independent model from the platform specific model from the system allowing it to run over multiple transports, on large and small (embedded). There are currently DDS API bindings for a host of languages including Ada, C, C++, C#, Java, Scala, Lua, Ruby and Node.js.

DDS can run over multiple transports including switch fabrics, UDP, TCP or shared memory. By leveraging a datagram service like UDP one can also take advantage of bandwidth optimizations such as multicast. This allows DDS to handle everything from command/control systems to video distribution. DDS also handles varied networks well, from sporadic wireless connections to high- performance switched fabrics. Systems that include some nodes with fast connections, some with slower connections, and even some with intermittent (e.g. wireless) connections will find DDS provides facilities to manage the resulting uneven delivery characteristics.

6.8.6.2 Entity Discovery

Entities in DDS are: Participants, Publishers, Subscribers, DataWriters and DataReaders. These entities are discovered and connected-to automatically, based on a discovery service built into DDS. This discovery service is implemented upon built-in topics that communicate which applications want to participate within a given domain and for each Participant, which topics they want to write and read (publish / subscribe). DDS does all the work of matching up like writers and readers to enable a fully flexible system that can be started up in any order, thus decoupling temporal attributes such as starting every node up at the same time. DDS however does not have a method of discovering new devices although there are several implementations that provide such functionality but are not standardized.

6.8.6.3 Quality of Service

QoS attributes can be applied for each individual Topic, Reader or Writer, as described in the following table:

ATTRIBUTE	REFERENCE [i.41]	DESCRIPTION
User Data	7.1.3.1	The purpose of this QoS is to allow the application to attach additional information to the created ***Entity*** objects such that when a remote application discovers their existence it can access that information and use it for its own purposes
Topic Data	7.1.3.2	The purpose of this QoS is to allow the application to attach additional information to the created ***Topic*** such that when a remote application discovers their existence it can examine the information and use it in an application-defined way
Group Data	7.1.3.3	The purpose of this QoS is to allow the application to attach additional information and be used by an application combination with the ***DataReaderListener*** and ***DataWriterListener*** to implement matching policies similar to those of the PARTITION QoS except the decision can be made based on an application- defined policy.
Durability	7.1.3.4	This QoS policy controls whether the Service will actually make data available to late-joining readers.
Presentation	7.1.3.5	This QoS policy controls the extent to which changes to data-instances can be made dependent on each other and also the kind of dependencies that can be propagated and maintained by the Service.
Deadline	7.1.3.7	This policy is useful for cases where a ***Topic*** is expected to have each instance updated periodically. On the publishing side this setting establishes a contract that the application must meet. On the subscribing side the setting establishes a minimum requirement for the remote publishers that are expected to supply the data values.
Latency Budget	7.1.3.8	This policy provides a means for the application to indicate to the middleware the “urgency” of the data-communication. By having a non-zero ***duration*** the Service can optimize its internal operation.
Ownership	7.1.3.9	This policy controls whether the Service allows multiple ***DataWriter*** objects to update the same instance (identified by ***Topic*** + ***key***) of a data-object.
Liveliness	7.1.3.11	This policy controls the mechanism and parameters used by the Service to ensure that particular entities on the network are still “alive.” The liveliness can also affect the ownership of a particular instance, as determined by the OWNERSHIP QoS policy.
Time_Based_Filter	7.1.3.12	This policy allows a ***DataReader*** to indicate that it does not necessarily want to see all values of each instance published under the ***Topic***. Rather, it wants to see at most one change every ***minimum_separation*** period.
Partition	7.1.3.13	This policy allows the introduction of a logical partition concept inside the ‘physical’ partition induced by a domain.
Reliability	7.1.3.14	This policy indicates the level of reliability requested by a ***DataReader*** or offered by a ***DataWriter***. These levels are ordered, BEST_EFFORT being lower than RELIABLE. A ***DataWriter*** offering a level is implicitly offering all levels below.
Transport Priority	7.1.3.15	The purpose of this QoS is to allow the application to take advantage of transports capable of sending messages with different priorities.
Lifespan	7.1.3.16	The purpose of this QoS is to avoid delivering “stale” data to the application.
Destination Order	7.1.3.17	This policy controls how each subscriber resolves the final value of a data instance that is written by multiple ***DataWriter*** objects (which may be associated with different ***Publisher*** objects) running on different nodes.
History	7.1.3.18	This policy controls the behaviour of the Service when the value of an instance changes before it is finally communicated to some of its existing ***DataReader*** entities.
Resource Limits	7.1.3.19	This policy controls the resources that the Service can use in order to meet the requirements imposed by the application and other QoS settings.

Table 6.8: Quality of Service attributes

6.8.6.4 Enhanced Data Typing

In addition to primitive types, DDS also provides support for Arrays, Multi-dimensional Arrays, Variable length Sequences, Enumerations, Unions, Value Types, Structure nesting and Opaque Byte arrays. By leveraging the ability to support multiple instances per data type by leveraging a Key, DDS allows for a more scalable system since multiple instances can be subscribed through a single topic. With the ratification of X-Types, DDS provides for extensible and mutable dynamic types allowing for data models to evolve while still preserving backward compatibility.

6.8.7 Protocol Stack

RTPS (Real-Time Publish Subscribe) defines the wire-format protocol for interoperable systems. It can be implemented over a transport service such as UDP or TCP.

```
 +--------------------------------+
|         DDS DCPS API           |
+--------------------------------+
|           Middleware           |
+--------------------------------+
|      DDSI-RTPS Wire Protocol   |
+--------------------------------+
|            Transport           |
+--------------------------------+
```

Figure 6.8.2: Protocol Layers

In the above figure, transport is defined as the means of providing error and flow control for reliable delivery. In this context it categorizes the set of protocols and mediums by which to transmit and receive application payload which can be TCP, UDP, Shared Memory, Wireless, High Performance switch fabrics, etc.

6.8.8 Data Model

Data within DDS can be as strictly defined or loosely defined as needed by the applications. A strict data model where each field is known to the middleware, enables the middleware to perform content-based filtering without the need for any application level programming to perform the filtering. In fact, in some implementations of DDS, these filters are discoverable and actuated on the publishing side so that only data that is relevant is pushed on to the network. For a loosely defined data model, there is the achievement of a very flexible architecture but at the cost of more data on the network to describe the data with schema.

6.8.9 Security

The OMG is currently in the process of creating a full Security specification for DDS that encompasses Authentication, Access Control, Encryption, Key Management and Auditing to be enforceable at a finer grain level of individual topics defined in the system. Different implementations provide various security features such as SSL encryption, X509 certificates and third party products help to implement policy and assurance.

6.8.10 Dependencies

DDS can be implemented on any underlying protocol as it provides all the necessary reliable communications as needed. Therefore it is common to have DDS run on UDP / TCP and other transports such as Serial links or even Shared Memory for communications between applications on the same node.

6.8.11 Benefits

The OMG Data Distribution Service for Real-Time Systems [i.41] is the only open standard for messaging that supports the unique Quality of Service (QoS) requirements of both enterprise and real-time systems. Scalability can be thought of as a Concave Function [i.53] and benchmarking would be highly implementation specific [i.54.] so we document the elements, which make DDS perform well in low latency and high throughput environments.

- Best-effort delivery mode which doesn't require acknowledgements
- Reduced message overhead since message headers and meta-data are sent on a per endpoint basis not per message which also increases throughput
- Arbitrary data-types reducing marshalling costs between user and middleware provided types
- Lightweight notification of data availability which helps to control filtering at the source
- Zero-copy data access removing multiple copy operations between middleware and application
- Brokerless peer-to-peer operations
- Incremental updates of field properties
- Multicast support

In summary, DDS direct peering model, data aware middleware and QOS control allows designers to build complex large scale systems which are robust under varying network conditions, context aware as state and QOS are part of each topic and easy to develop with common language bindings and tool chains. **Applicability to Service Oriented Architecture:** Service Oriented Architecture deals with developing software as a series of interoperable services, which are decoupled but maintain a common representation of distributed resources. SOA can be accomplished through many approaches such as DCOM, DDS, CORBA, Java RMI and of course Web Services. It is Web Services that have a more ubiquitous view of representation (XML, JSON) and identification (URI). DDS is currently exploring examples with several pre-standard implementations currently in development [i.55].

6.9 Modbus Protocol

The following clauses describe the Modbus Protocol.

6.9.1 Background

Modbus was first introduced by Modicon (now part of Schneider Electric) for process control systems. It was used as a master-slave protocol for serial communication interfaces (such as RS232 / RS485). Versions of Modbus protocol now also exist for TCP/IP/Ethernet.

6.9.2 Status

Evolution of this protocol is managed by the Modbus.org community.

- "Modbus application protocol specification" specifies Modbus application layer protocol that is positioned at layer 7 of the OSI model.
- "Modbus over serial line specification and implementation guide" deals with Modbus master / slave protocol (for RS 232 / RS 485) that is positioned at layer 2
- "Modbus messaging on TCP/IP implementation guide" provides information that helps developers implement the Modbus messaging service.

The Modbus protocol was transferred from Schneider Electric to the Modbus Organization in April 2004. The specification is available free of charge for download, and there are no subsequent licensing fees required for using Modbus or Modbus TCP/IP protocols. Additional sample code, implementation examples, and diagnostics are available as part of the Modbus TCP toolkit, available free of charge to Modbus Organization members and available for purchase by non-members.

6.9.3 Category and Architectural Style

Modbus is an application layer messaging protocol and provides client-server communication between devices connected on different types of buses and networks. Some of the supported buses and networks include the following (as shown in Figure 6.9.1):

- Serial communication over EIA/TIA-232, EIA/TIA-485
- TCP/IP/Ethernet
- A high speed token passing network (Modbus Plus)

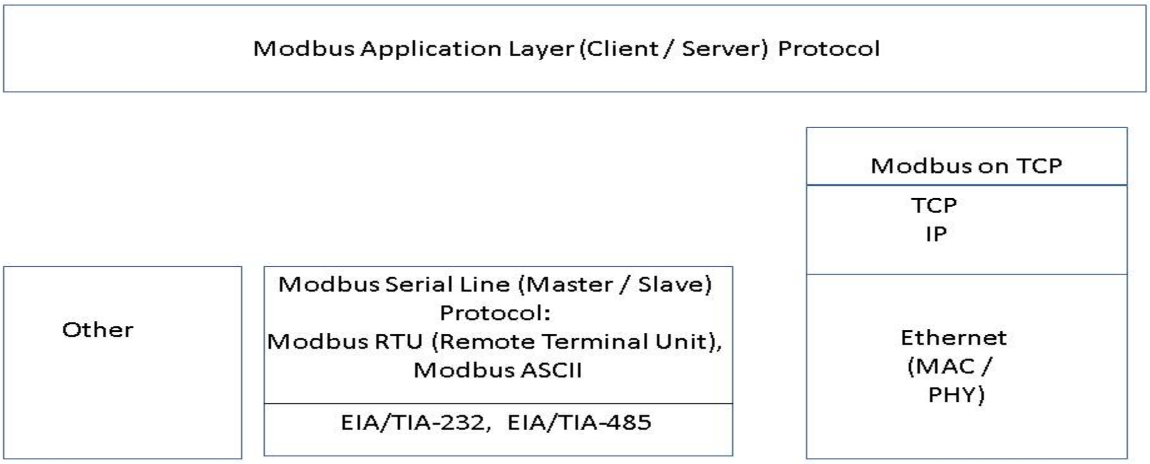


Figure 6.9.1: Modbus for serial communication and TCP/IP/Ethernet Modes

An example network architecture is shown in Figure 6.9.2.

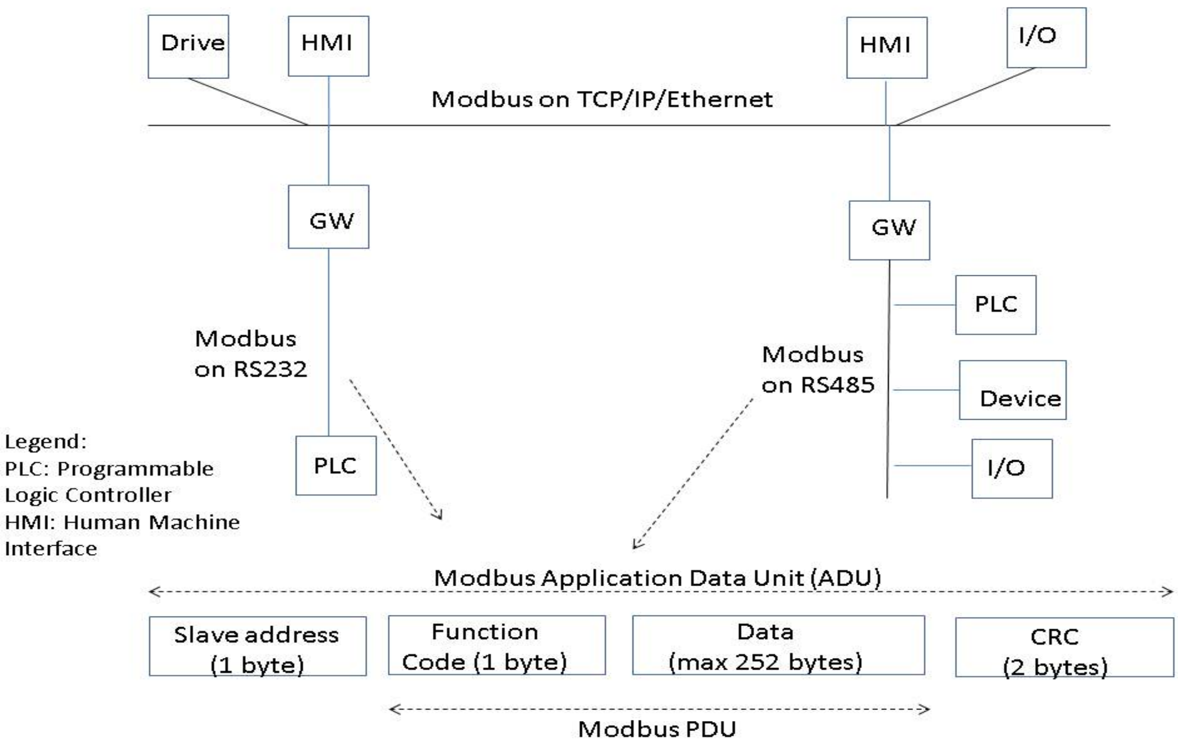


Figure 6.9.2: Example - Modbus Network Architecture

6.9.4 Intended use

Examples of intended uses for Modbus are:

- Process measurement and control applications in automation industry
- Building automation. Power substation applications.
- Master – slave applications in industrial environment to monitor and program devices. To transfer discrete / analog I/O and register data between control devices.
- To monitor and control field devices using PC. For example, to connect a supervisory computer with an RTU (Remote Terminal Unit) in a SCADA (Supervisory Control and Data Acquisition System)

6.9.5 Deployment Trend

There are several organizations that supply Modbus solutions. These are listed at:
http://www.modbus.org/companies.php

http://www.modbus.org/faq.php states that industry analysts have reported over 7 million Modbus nodes in North America and Europe alone.

6.9.6 Key features

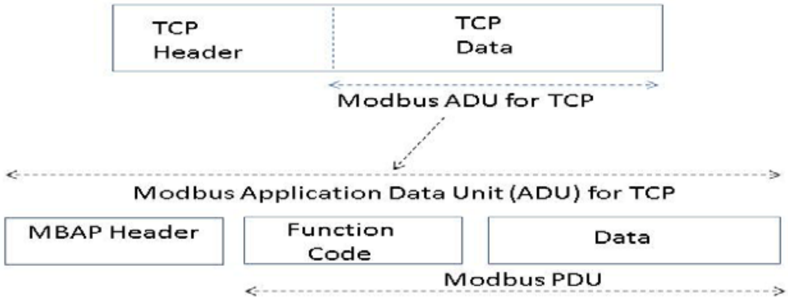
Modbus uses client / server model and supports devices communicating with serial interfaces (such as RS232/RS485) as well as devices with TCP/IP/Ethernet.

6.9.7 Protocol Stack

Modbus supports a requests / response pattern. Modbus for serial communication supports one master and a maximum 247 slaves for Modbus version that runs over serial interfaces. Master issues a unicast request and slave responds to that. Modbus also supports broadcast mode where master's request is sent to all the slaves but no slave responds to broadcast request. A Modbus frame or Modbus Application Data Unit (ADU) consists of the following:

- Additional address field: A field containing additional addresses used by the underlying communication protocol. It is 1 byte slave address for Modus master / slave protocol that runs over serial links (such as RS 232, RS 485). For Modbus-TCP, it is called Modbus Application Protocol (MBAP).
- Modbus PDU: It is independent of underlying communication layer and consists of two parts: 1) 1-byte Function code to indicate identity of the requested service, 2) Variable length data field containing payload of the requested service.
    - There are three types of Modbus PDUs: Modbus Request, Modbus Response and Modbus Exception.
- An optional error check field.

For Modbus communication over a serial interface, maximum ADU size for RS485 is 256 bytes and maximum PDU size is 253 bytes as shown above in Figure 6.9.2. Maximum PDU size for Modbus TCP is also restricted due to restriction for Modbus PDU size for serial communication. Error check field of Modbus ADU is not used as TCP already uses CRC. MBAP is 7 bytes for Modbus TCP as shown in Figure 6.9.3. Modbus TCP/IP can be accessed at port 502.



Modbus Application Protocol (MBAP) Header	
Transaction identifier (2 bytes)	This is used as a salve can handle multiple requests with Modbus-TCP
Protocol identifier	Set to 0 for Modbus.
Length	To help recognize message boundaries
Unit Identifier (1-byte)	Not used for Modbus-TCP as slave identified by its IP address. If remote device is on a non-TCP network, that is identified by this field.

Figure 6.9.3: Modbus - TCP

6.9.8 Data Model

The Modbus standard defines bit-addressable and 16-bit word addressable input and output data items. An example of Modbus data types is shown in Figure 6.9.4.

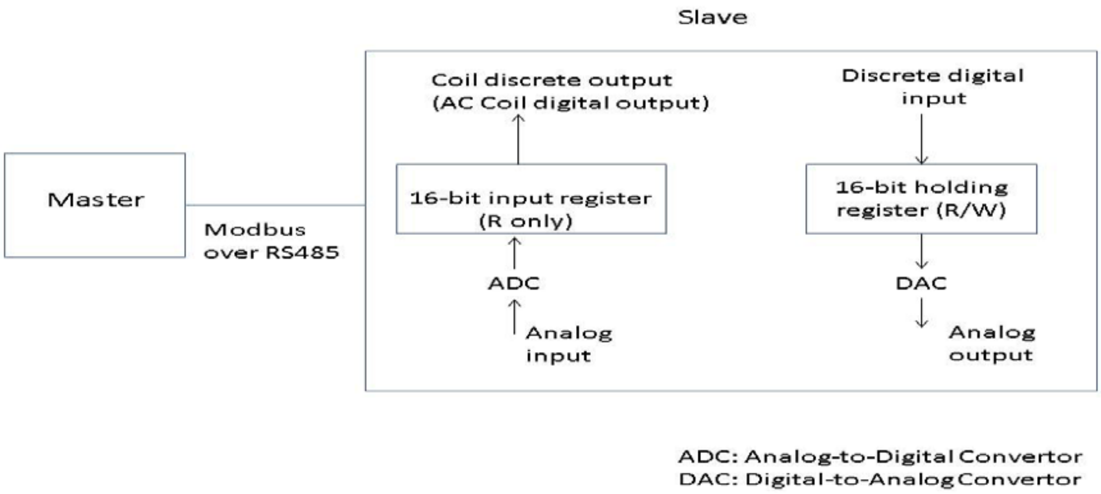


Figure 6.9.4: An example of Modbus data types

- Modbus data on a device is stored in a series of tables. Following are the four primary tables specified:
- Two table types for "single bit" object type:
  - Physical Discrete input to read status of discrete inputs in a remote device. Up to 2000 contiguous bits can be read at a time.
  - Coils table type for read and write.
- Two table types for "16-bit word" object type:
  - Input Registers to read up to 125 contiguous input registers from a remote device
  - Holding Registers (for read / write)

Each table allows up to 65,535 data items. These tables may be overlapped or mapped to separate blocks of memory as shown in Figure 6.9.5. Modbus uses Big Endian representation for address and data fields. Modbus also supports concept of a file where file is an organization of records.

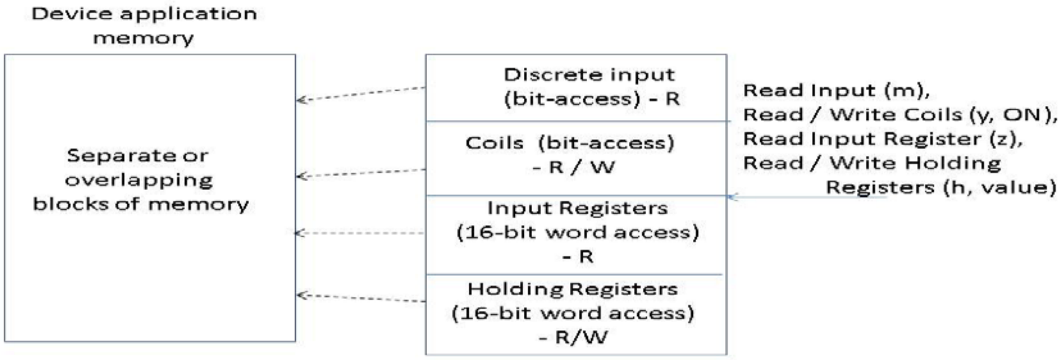


Figure 6.9.5: Modbus data model

6.9.9 Security

Modbus does not provide explicit security mechanisms (such as mutual authentication of Modbus master – slave, encryption, integrity protection, access control....) on its own and relies on other mechanisms for this purpose.

6.9.10 Dependencies

Modbus for serial communication runs over serial interfaces such as RS-232 and RS-485. Modus-TCP is dependent on TCP and TLS.

6.9.11 Benefits and Constraints

6.9.11.1 Benefits

Modbus benefits include:

- Open standard.
- Lightweight protocol for industrial communication over serial links
- Large deployment for industrial applications
- Can work over serial links (RS 232 / RS 485) as well as over TCP/IP/Ethernet.
- CIP (Common Industrial Protocol) -Modbus integration performs translations to make Modbus device data consistent with CIP model

6.9.11.2 Constraints

Some Modbus constraints include:

- Only polling mode (request / response) supported for Modbus over serial interfaces.
- No discovery or advertisement mechanisms supported

- Small PDU size
- As with some other protocols, Modbus doesn't provide an explicit security mechanism on its own and relies on other mechanisms for this purpose.

6.9.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by the Modbus Protocol is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of the overall M2M system and Modbus is one component of this system. For example, Modbus relies on the security mechanisms supported by other layers in the network where it is deployed. To specifically compare with each requirement, one would need to take several assumptions about other system components and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.9.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-008, OSR-010, OSR-024, OSR-028, NFR-002

6.9.12.2 Partially Supported Requirements

OSR-001, OSR-005, OSR-006, OSR-007, OSR-009, OSR-011, OSR-013, OSR-015, OSR-016, OSR-017, OSR-019, OSR-020, OSR-029, OSR-030, OSR-031, OSR-037, OSR-038, OSR-040

NOTE: OSR-001 is shown in clause 6.9.12.1 (Fully) as well as 6.9.12.2 (Partially). ModBus is deployed over serial interfaces such as RS-485, but also supports deployment over IP based protocols.

6.9.12.3 Unsupported Requirements

OSR-018
(This requirement is for cellular devices)

SER-004 to SER-006
(These requirements are for UICC based devices. Enhancements would be needed for support of UICC based devices with Modbus systems)

6.10 DNP3 Protocol

The following clauses describe the DNP3 (Distributed Network Protocol - 3).

6.10.1 Background

DNP was originally developed by Westronic, Inc. (now GE-Harris) for electric utility industry. It was later released to DNP user group (www.dnp.org) for further development. In 2010, DNP Technical Committee worked with IEEE to establish DNP3 as an IEEE standard and IEEE 1815TM-2010 was released that year. An updated standard, IEEE 1815TM-2012, was released in 2012.

6.10.2 Status

DNP3 is currently an IEEE standard.

- IEEE Standards for Electric Power Systems Communications – Distributed Network Protocol (DNP3), IEEE Std 1815TM-2012 [i.43] (obsoleted IEEE Std 1815TM 2010)
- IEEE1815 has been accepted in the NIST catalog of smart grid standards.
- DNP user group continues to work on this.

6.10.3 Category and Architectural Style

DNP3 supports client – server model between DNP3 master and DNP3 outstation. Master contacts outstation to read some data or carry out a control command or for some other operation. An IED (or RTU) is an example of an outstation. Some examples of IEDs include sensors, meters, actuators, PLCs and accumulators. An RTU could be managing multiple IEDs though RTU also appears as IED to master when DNP3 is used. Some possible deployment scenarios of DNP3 are shown in Figure 6.10.1. Different layers of DNP3 stack are shown in Figure 6.10.2. User software makes use of DNP3 application layer software to send and receive messages. Some of the supported data types by DNP3 include binary, analog and counter data types. Each (master or outstation) device is given a 16-bit address that is unique in the context of devices that are addressed by a master (on a link).

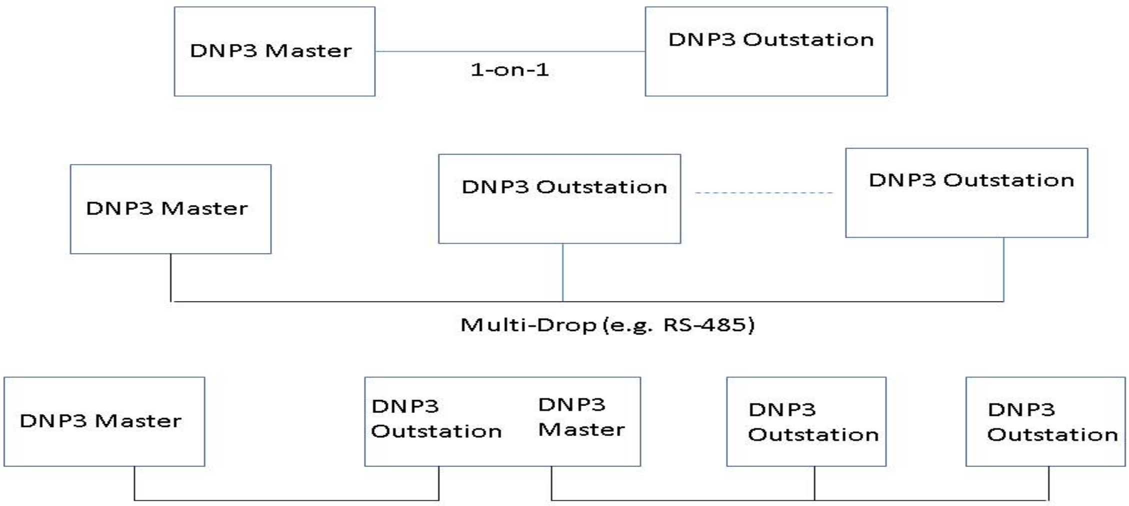


Figure 6.10.1: Example DNP3 deployment scenarios

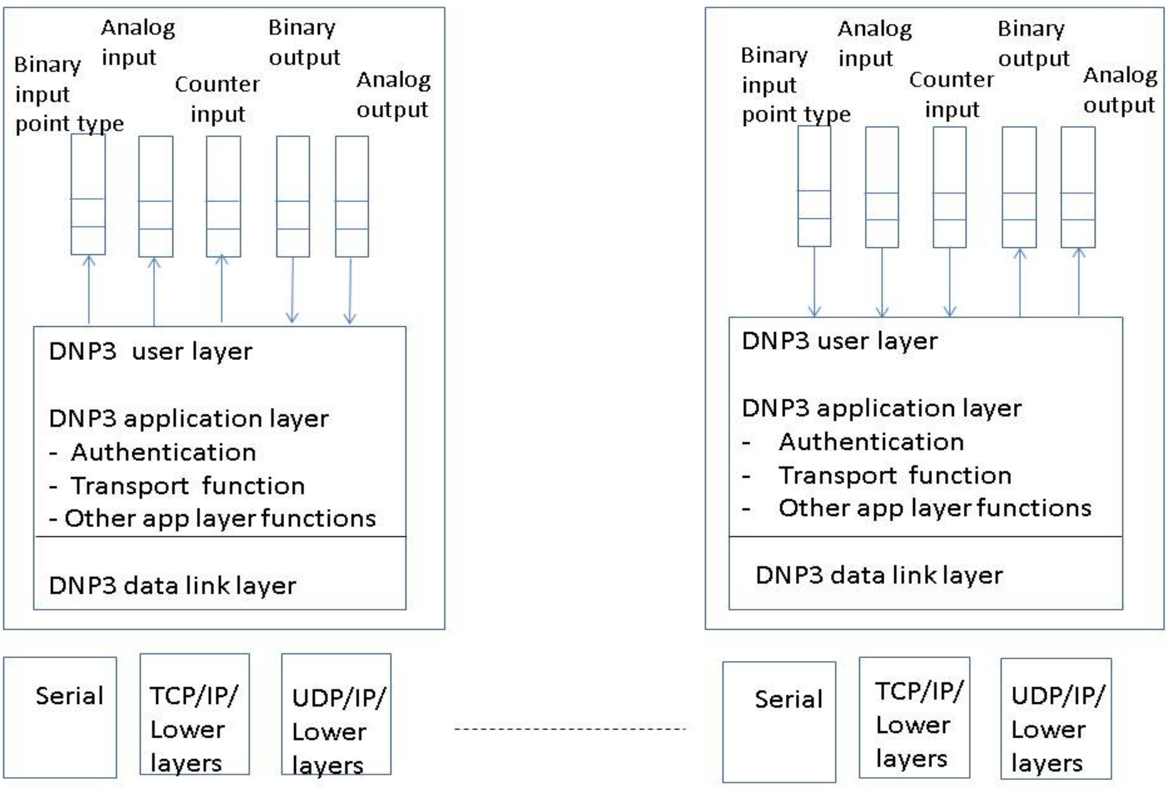


Figure 6.10.2: DNP3 Master and Outstation

6.10.4 Intended use

DNP3 is intended for:

- communication between SCADA master station and IEDs or RTUs
- use in electric and water utility industries. For example, DNP3 can be used to allow a computer in the operations center of an electric utility company to communicate with computers located in substations. A substation has devices such as current sensors, circuit breakers, voltage transducers, temperature sensors, surveillance cameras, water level monitors etc. that need to be monitored and/or controlled.
- also, for other industry segments in the oil and gas industry.

6.10.5 Deployment Trend

DNP3 is used by utilities such as electric and water companies for communication between data acquisition and control equipment. As referenced in http://www.dnp.org/Lists/Announcements/Attachments/6/Newton%20Evans%20NA%20SSA%202011V1.pdf, it is being used for the following:

- Communication within a substation
- Substation to substation communication
- Substation to external host communication

As shown in http://csrc.nist.gov/cyberframework/rfi_comments/west_030713.pdf, DNP3 is used by approximately three-quarters of North American electric utilities. It also states that DNP3 is being adopted by an increasing number of water and wastewater utilities, and is used in oil & gas and other SCADA applications.

A list of companies that provide DNP3 products is given at http://www.dnp.org/Pages/DnpProductsDefault.aspx

6.10.6 Key features

Some key features of DNP3 are:

- Request / response model with multiple data types in the same message.
- Supports unsolicited mode where an outstation (or a slave) node can send unsolicited messages to master node
- Report-by-exception is supported where only changes are reported by an outstation.
- Provides reliability at link layer for lossy links by providing error detection, optional acknowledgement feature for data link layer frames, detection of duplicate frames, , 2-byte CRC for data link layer header and 2-byte CRC for every 16 bytes of data in the data part of data link frame.
- Provides fragmentation and re-assembly at pseudo transport layer (that is part of application layer).
- Supports a time synchronization mechanism between master and outstation (especially for DNP3 over serial links). Uses a standard format for this.
- Supports time related activities (e.g. an IED to do a specific activity at a certain time, an IED to add time stamps to events).
- Optimizes transmission of data acquisition and control commands in SCADA systems
- Supports serial communication (such as using RS232 / RS485) as well as TCP/IP and UDP/IP based protocols.

6.10.7 Protocol Stack

As shown above in Figure 6.10.2, the DNP3 stack supports following layers:

- DNP3 user software makes use of DNP3 application layer software to send and receive messages. It also interacts with database on that device.

- DNP3 Application layer: DNP3 outstation (and/or master) device may have limitation on the maximum size for an application message. For example, this limitation could be due to limited memory in an RTU or IED. DNP3 application layer breaks down large application layer messages into multiple fragments as needed. A master is expected to be ready to receive fragment size of 2048 octets or above. An outstation may limit the fragment size to much smaller value. An application level acknowledgement feature is supported by which a receiver can confirm receipt of an application fragment. A sequence number field is used to detect duplicate application fragments.

    - DNP3 transport function is part of DNP3 application layer. It supports segmentation and re-assembly of DNP3 application layer fragments (to / from link layer frames).

    - Authentication feature is also supported as part of DNP3 application layer.

- DNP3 link layer provides reliable link layer. Maximum length of link layer frame is 292 bytes.

A high level view of DNP3 packet processing is given in Figure 6.10.3.

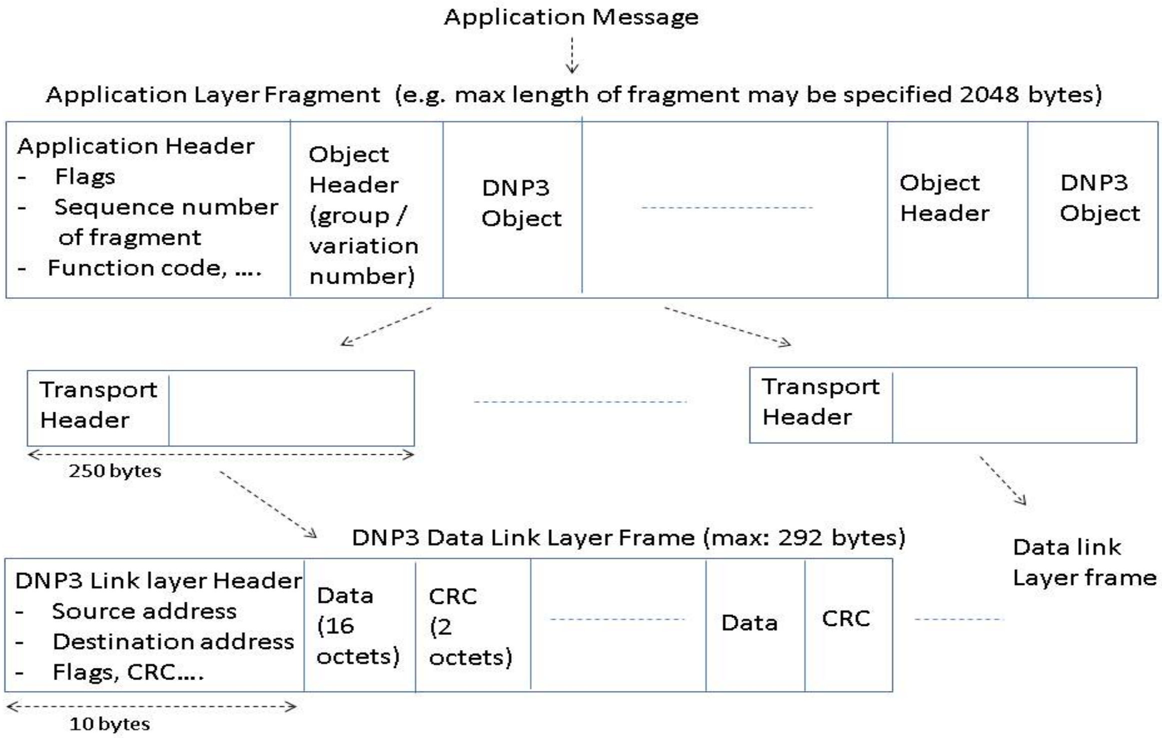


Figure 6.10.3: DNP3 Protocol – Packet Processing (High level view)

Some examples of operations that DNP3 master can carry-out on an outstation (by using a suitable function code in the application fragment header) are given here:

- Read data
- Set time on an outstation (as part of time synchronization)
- Send requests for control operations
- Set analog output values
- Freeze accumulator requests
- Cold restart
- Warm restart

- Freeze and clear some counters
- Select-before-operate
- Direct operate
- Start or stop running an application

6.10.8 Data Model

DNP3 data types are conceptually organized as array of points where a point is a uniquely identifiable entity. Following are some of the point types used by DNP3:

- Binary input: Device state (such as state of circuit breaker: closed or tripped) is stored in an array of Boolean values.
- Analog input: Input quantities measured or computed by outstation
- Counter input (such as Kilowatt hours)
- Binary output (e.g. ON/OFF)
- Analog output

DNP3 can also transport files and other data types. DNP3 uses index numbers (such as 0, 1, 2,….) to identify points in a point array. It has provisions to represent data in different formats. A "group number" is used to classify type of data within a message and a "variation" is used to indicate encoding format. Index and group number identify a unique point. Value of a point may represent current value (called static data in DNP3) or an event. For example, an event could be triggered if a binary value changes (e.g. from ON to OFF) or when an analog value crosses its threshold. An example is shown below:

- Current value of analog input point : (object) group 30
- Event analog value: (object) group 32
- Some variations (or encoding choices) available with (object) group 30:

  1: 32 bit integer value with flag

  2: 16 bit integer value with flag

  3: 32 bit integer value

  4: 16 bit integer value

  5: 32 bit floating value with flag

  6: 64 bit floating value with flag

6.10.9 Security

One way, as well as mutual authentication, between DNP3 master and outstation at application layer is supported. If an outstation receives a message from master (such as set some critical variable), it can use challenge / response mechanism to authenticate the master before processing that message. Use of pre-shared keys is allowed for authentication. Function codes and variations (i.e. encoding methods) are defined for authenticated messages. Optional support for public key cryptography has been also added.

6.10.10 Dependencies

Dependencies for DNP3 include:

- Depends on configuration of shared key in master and outstation if shared key method is used for authentication (and on public key cryptography mechanisms if that option is used for authentication).

- Need to use external protocols if DNP3 data is to be encrypted.

6.10.11 Benefits and Constraints

6.10.11.1 Benefits

Benefits of DNP3 include:

- Open standard. An IEEE standard since 2010.
- One of the common protocol used in the SCADA systems especially electric utility segment
- Can work over serial links (such as RS 232 / RS 485) as well as over TCP/IP (or UDP/IP).

6.10.11.2 Constraints

As with some other IoT protocols, encryption is not explicitly included, and DNP3 relies on other mechanisms for that purpose.

6.10.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by DNP3 is shown in the following clauses:

NOTE: Many requirements from TS-0002 [i.2] depend on the architecture of overall M2M system and DNP3 comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.10.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-004, OSR-008, OSR-010, OSR-014, OSR-019, OSR-024, OSR-028, OSR-040, OSR-058, OSR-060, OSR-061, SER-002, SER-003, SER-009.

6.10.12.2 Partially Supported Requirements

OSR-005, OSR-006, OSR-007, OSR-009, OSR-011, OSR-012, OSR-013, OSR-015, OSR-016, OSR-017, OSR-020, OSR-029, OSR-030, OSR-031, OSR-037, OSR-038, OSR-042.

6.10.12.3 Unsupported Requirements

OSR-018
(This requirement is for cellular devices)

SER-004 to SER-006
(These requirements are for UICC based devices. Some enhancements would be needed for support of UICC based devices by DNP3)

6.11 UPnP Cloud

The following clauses describe the UPnP (Universal Plug and Play) Cloud Protocol. [i.48]

6.11.1 Background

Formed in October 1999, the UPnP Forum is an industry initiative of more than 1000 leading companies in computing, printing and networking; consumer electronics; home appliances, automation, control and security; and mobile products.

Goals

The Forum's goals are to allow devices to connect seamlessly and to simplify network implementation in the home, corporate and cloud environments. Toward this end, UPnP Forum members work together to define and publish UPnP device control protocols built upon open, Internet-based communication standards.

Leadership

A member-based Steering Committee provides Forum leadership and business direction, while several technical working committees identify and define UPnP devices, services, protocols and usage scenarios.

The UPnP architecture offers pervasive peer-to-peer network connectivity of PCs of all form factors, intelligent appliances, and wireless devices. The UPnP architecture is a distributed, open networking architecture that leverages TCP/IP and the Web to enable seamless proximity networking in addition to control and data transfer among networked devices in the home and away from home by means of the cloud extensions.

Standardized Device Control Protocols

UPnP standards are based upon Device Control Protocols (DCPs), which are the domain specifications based on the UPnP Device Architecture. The DCPs specifications are based on protocols that are: declarative, expressed in XML and communicated via HTTP. For more information on UPnP technology see [i.48]

In December 2008 UPnP Device Architecture Version 1.0 and seventy-two (72) UPnP Device Control Protocols specifications were adopted and published by the International Standards Organization (ISO) and International Electrotechnical Commission (IEC) as International Standards. See [i.51]. In the fall of 2011 UPnP Device Architecture Version 1.1 and an additional twenty-one (21) UPnP Device Control Protocols specifications were newly adopted and published by the International Standards Organization (ISO) and International Electrotechnical Commission (IEC) as International Standards in the fall of 2011. Eight (8) specifications were also published as updates to previously published ISO/IEC International Standards. See [i.49] and [i.50].

UPnP technology targets wide area networks, home networks, proximity networks and networks in small businesses, commercial buildings and is agnostic over different connected networks. It enables data communication between any two devices under the command of any control device on the network. UPnP technology is independent of any particular operating system, programming language, or network technology.

6.11.2 Status

The following list includes the specifications, which have been standardized. The standardization process includes obtaining three sample implementations of the Device Control Protocol (DCP) to pass the UPnP Certification Test Tool, circulating the specification for a mandatory Forum member review and comment period, and obtaining the approval of the Steering Committee to become a Standardized DCP. Standardized DCPs are available to the public. The UPnP forum has a rigorous certification program, based on (low cost) self-certification. Certifications exist for the device and the control point side for each DCP.

Published Device Categories (DCPs) listing only the latest version:

- Audio/Video, Audio and Video transport and control, schedule recording.
    - MediaServer:4 and MediaRenderer:3
    - ContentSync:1
- Device Management, including configuration, software management, including transport testing
    - Manageable Device:2
    - BasicManagementService:2
    - ConfigurationManagementService:2
    - SoftwareManagementService:2

- EnergyManagement:1
- Low Power:1

- Home Automation, various home automation protocols
  - SolarProtectionBlind:1
  - Digital Security Camera:1
  - HVAC:1
  - Lighting Controls:1
  - SensorManagement:1
  - DataStore:1
- Networking, routing functionality.
  - Internet Gateway:2
  - WLAN Access Point:1
- Printer, document and photo printing and scanning.
  - Printer Enhanced:1
  - Printer Basic:1
  - Scanner:1
- Remoting, mechanism to detect and setup remote UIs, like VNC.
  - Remote UI Client:1 and Remote UI Server:1
- Telephony, access and distribution telephony (2 way communication).
  - Telephony:2
- Add-on Services, generic services that can be used in any context.
  - DeviceProtection:1, adding orthogonal access control to all DCPs
  - Quality of Service:3, QOS streaming

In mid-2013, a UPnP Cloud Task Force was created. Its objectives were to:

- Maintain UPnP behaviour—it must still just work!
- Cloud-enable all existing UPnP specifications (and existing devices)—adapt & adopt, not re-invent
- Introduce user-specific capabilities
- Facilitate the always-connected lifestyle

This task force has the charter to make an UPnP Cloud profile as add-on mechanism to the existing UPnP Device Architecture. The charter contains a statement that the existing DCPs needs to be leveraged to the cloud; only changes on UPnP Device Architecture (UDA) level are allowed. This means that all domain knowledge in the DCPs will be enabled towards the cloud. The task force is finalizing the UPnP Cloud Architecture (UCA), which will be published in Q1 of 2014. Current efforts are ongoing to specify a certification program.

6.11.3 Category and Architectural Style

UPnP is a client server model, where the server is denoted as UPnP Device and a client as Control Point (CP).

There are many different kind of UPnP devices already standardized, but they all share a common framework that is denoted as UPnP Device Architecture (UDA). The UDA is described as a framework, which is not specific to any domain. The UDA describes the components for discovery, description, action handling and eventing. This Framework layer is used for describing the domain specific Device Control Protocol (DCP).The device architecture is published at [i.52].

UPnP DCPs are expressed in constructs supplied by the Framework described by the UDA. The DCPs are expressed in declared state variables that can be evented, and RPC functions. For more information on UPnP technologies see [i.48]

UPnP Cloud has a similar architecture but replaces (adds for cloud access) the discovery and eventing and uses XMPP as transport mechanism to relay the same information as described in any DCP.

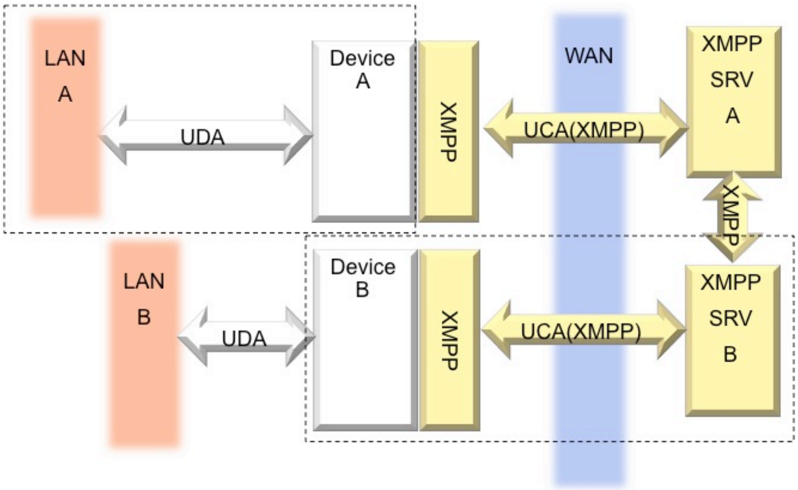


Figure 6.11.1 – Architecture: XMPP added to UPnP

6.11.4 Intended use

UPnP Cloud is intended to be used for P2P, P2M and M2M / IoT purposes.

6.11.5 Deployment Trend

The UPnP specifications are implemented in a broad array of available server and client software stacks, including many open source options. The UPnP forum is being supported by an extensive worldwide developer community.

UPnP is deployed in billions of installed devices.

AV DCPs deployment examples are:

- Every Windows PC since Windows ME
- Every PS3 and Xbox 360
- Most connected TVs, Blu-ray players, and smart phones
- Every media NAS

IGD DCPs deployment examples are:

- Every home router
- Every Wi-Fi device with Wi-Fi Protected Setup

UPnP specifications are referenced for example by: W3C, Wi-Fi Alliance, DLNA and the Connected Car Consortium (CCC).

The first deployments of UPnP Cloud are expected in 2014.

6.11.6 Key features

Key features of UPnP include:

- Separation of concern by separation of the used transport mechanisms (UDA/UCA) and domain specific knowledge in the device descriptions (DCPs).
- UPnP Device Architecture provides mechanisms for automatic discovery of UPnP devices and self-describing of the capability of the detected devices. The UPnP devices expose services which the UPnP control points can use to fulfil a function. The collection of services describes which domain specific actions and state variables are implemented. Hence each Domain (DCP) is has its own described set of actions and state variables. The transport mechanisms of conveying the DCP prescribed information are expressed in the UDA [i.52] and UCA specification documents. [i.52]. The UPnP architecture supports zero-configuration and automatic discovery whereby a device can:

  a. Dynamically join a network

  b. Obtain an IP address

  c. Announce its name

  d. Convey its capabilities upon request
  Described in device and service descriptions

  e. Learn about the presence and capabilities of other devices

  f. Leave a network smoothly and automatically without leaving any unwanted state information behind

- The UPnP Device Control Protocol Specifications (DCPs)
  The domain specific DCPs capabilities are expressed by means of one or more services, each service containing:

  a. Set of actions
  Input and output arguments of each action are typecast by state variables

  b. Set of state variables
  State variables are statically typed.
  Basic types such as Boolean, integer float can exist.
  Complex types like structs are modelled in XML and defined by XSD schemas
  State variables indicated as evented are delivered asynchronously.

  c. Relationships between actions and events.

- The UPnP specifications are backwards compatible, and are extensible for vendors (and other standards organizations).
- UPnP is extended into the cloud by using an existing proven standard XMPP while maintaining the UPnP and XMPP benefits. Combining these 2 widely used standards will lead to new propositions in the market. The transport mechanisms of SSDP and GENA (see 6.x.7) then are replaced by standard XMPP constructs like presence and pub-sub.

  a. The mechanisms for local and cloud access differ, but maps the same functionality as described in the domain specific DCP. Hence all UPnP DCPs will work with the UPnP Cloud Architecture.

  b. XMPP works from inside a browse by using BOSH or web sockets; hence all UPnP described DCPs works from inside a web browser.

  c. XMPP cloud infrastructure is scalable.

  d. XMPP exchange is secure
  Uses SASL for authentication.
  Uses TLS for secure connections.

6.11.7 Protocol Stack

The common parts of an UPnP stack are standard internet technologies like:

- HTTP (Hyper Text Transfer Protocol) is being used as transport layer for SOAP/GENA and device and service descriptions.
- SSDP (Simple Service Discovery Protocol) is being used to detect UPnP devices on the network.
- SOAP (Simple Object Access Protocol) is being used to invoke UPnP actions.
- GENA (Generic Event Notification Architecture) is being used to event state changes.

UPnP vendor			
UPnP Forum			
UPnP Device Architecture			
SSDP equivalent mapped to XMPP presence	**Multicast events equivalent mapped to XMPP "pubsub"**	SOAP mapped to XMPP SOAP support	**GENA equivalent mapped to XMPP "PubSub"**
XMPP			
TLS/SASL			
TCP			
IP			

Figure 6.11.2: – UPnP protocol stack

The UDA abstracts and unifies the SSDP, SOAP, GENA and multicast events in a single framework. SSDP is being used to convey the location of the device description. The device description document (DDD) than can be retrieved by means of HTTP. The device description contains information about the device and the implemented services in the device. The service description location can be derived from the information in the device description and can also be retrieved by means of HTTP. The service control protocol document (SCPD) is a list of SOAP action and state variables, describing the functionality of the UPnP device.

UPnP Cloud consists of using XMPP stanzas to convey the UPnP information. Mapping of UDA constructs on UCA constructs is shown in the table below.

	Addressing	Discovery	Description	Eventing	Control	Presentation
UDA	Static/ DHCP/ AutoIP	SSDP	DDD/SCPD	GENA	SOAP	HTML
UCA	XMPP Full JID from User	XMPP Presence	XMPP <iq> + disco#info	XMPP pubsub	SOAP over XMPP	TBD

	ID + UDN		cap xchng				
Ref	RFC 6120 - XMPP CORE [i.6] RFC 6121 - XMPP IM [i.14]	RFC 6120 - XMPP CORE [i.6] RFC 6121 - XMPP IM [i.14]	XEP-0030 [i.19] XEP-0127 [i.44] XEP-0115 [i.45]	XEP-0060 [i.21] XEP-0248 [i.46]	XEP-0072 [i.47]		

Table 6.11: UPnP Device Architecture (UDA) + XMPP = UPnP Cloud Architecture (UCA

6.11.8 Data Model

Each DCP has its own communication model based on top of the UPnP Architecture.

The domain is modelled by means of state variables and actions.

Each domain has its own set of state variables and actions.

Data between the UPnP device and control points are being conveyed by actions in and input and output arguments of each action are type cast as a state variable. .

State changes of the UPnP devices are evented, the evented data is described and typecasted by state variables. State variables can be of simple or complex types. Complex types are expressed in XML and are defined by an XSD schema.

6.11.9 Security

UPnP has an add-on service called Device Protection.

This service allows using secure connections for invoking actions by means of HTTPS (TLS).

Cloud enabled UPnP has same protection mechanisms as XMPP, using TLS as encryption of the channel and SASL for authentication.

6.11.10 Dependencies

The following dependencies are noted for UPnP Cloud:

- Cloud enabled UPnP uses XMPP [i.6] as transport layer.
- XMPP streams as defined in RFC6120 [i.6] use TCP as transport
- Use of HTTP [i.7] as transport is allowed as per XEP-0124 [i.32] and XEP-0206 [i.33]
- Uses TLS [i.16] and SASL [i.17] for security.
- Jingle [i.27] extensions use XMPP for signalling but data plane packets is sent over other transport mechanisms such as TCP or UDP [i.10]
- Uses XML [i.12] for defining messages.

6.11.11 Benefits and Constraints

6.11.11.1 Benefits

- **Media and device independence.** UPnP technology can run on any network technology including Wi-Fi, coax, phone line, power line, Ethernet and 1394.
- **Platform independence.** Vendors can use any operating system and any programming language to build UPnP products.
- **Internet-based technologies.** UPnP technology is built upon IP, TCP, UDP, HTTP, XML and XMPP among others.
- **UI Control.** UPnP architecture enables vendor control over device user interface and interaction using the browser.
- **Programmatic control.** UPnP architecture enables application programmatic control.
- **Common base protocols.** Vendors agree on base protocol sets on a per-device basis.
- **Extendable.** Each UPnP product can have value-added services layered on top of the basic device architecture by the individual manufacturers.
- **UPnP is easily extensible.** It provides basic set of features that can be expanded by protocol extensions to provide new set of features. This can be either new DCPs or new versions of a DCP. Also due to the self-describing nature vendor specific extensions are possible
- **UPnP Cloud is based on XMPP.** This provides an existing cloud infrastructure with proven track record. See benefits of XMPP, clause 6.5.11.1.
- **UPnP control points can interact with all UPnP devices.** Due to the nature of UPnP each control point can talk directly 1-1 to one specific UPnP device in a P2P like manner, but they can talk to many devices simultaneously. Also UPnP control points and UPnP devices can co-exist in one physical device (aka a physical box), this depends on the domain specific requirements.
- **UPnP bridges other network technologies.** UPnP in the home can be used to bridge various different kinds of sensor networks. This is described in the Sensor Management specifications.
- **UPnP can communicate in the home without having connection to the internet;** hence when access to the internet is severed, all local devices can still work together by means of the UDA to perform the desired functionality, this means that sensor data collection still can take place and can be uploaded to the internet based server when the internet connection is restored.

6.11.11.2 Constraints

- Use of TCP may not be desirable for some IoT segments.
- Overhead may be high for XML data descriptions conveyed by SOAP messages.

6.11.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by uPnP Cloud is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and uPnP Cloud comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.11.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-005, OSR-006, OSR-007, OSR-008, OSR-009, OSR-010, OSR-011, OSR-012, OSR-014, OSR-015, OSR-016, OSR-017, OSR-018, OSR-019, OSR-021, OSR-022, OSR-023, OSR-024, OSR-025, OSR-026, OSR-027, OSR-028, OSR-029, OSR-030, OSR-034, OSR-035, OSR-037, OSR-038, OSR-039, OSR-041, OSR-043, OSR-044, OSR-046, OSR-049, OSR-051, OSR-053, OSR-054, OSR-055, OSR-056, OSR-057, OSR-058, OSR-059, OSR-060, OSR-061, OSR-062, OSR-063, OSR-065, OSR-066, OSR-070, OSR-071

MGR-001, MGR-008, MGR-009, MGR-010, MGR-012, MGR-013, MGR-014, MGR-015,

ABR-001, ABR-002, ABR-003

SMR-001, SMR-002, SMR-003, SMR-004, SMR-005, SMR-006, SMR-007

SER-002, SER-003, SER-004, SER-007, SER-011, SER-019, SER-020, SER-022, SER-025

OPR-001, OPR-002, OPR-003,

CRPR-001, CRPR-002, CRPR-004, CRPR-005

6.11.12.2 Partially Supported Requirements

OSR-004, OSR-013, OSR-020, OSR-031, OSR-032,OSR-033, OSR-036, OSR-040, OSR-042, OSR-045, OSR-047, OSR-048, OSR-052, OSR-064, OSR-067, OSR-068, OSR-069, OSR-072

MGR-002, MGR-003, MGR-004, MGR-005, MGR-006, MGR-011, MGR-016, MGR-017,

SER-001, SER-005, SER-006, SER-008, SER-009, SER-010, SER-012, SER-018, SER-021, SER-023, SER-024, SER-026,

CHG-001, CHG-002, CHG-003, CHG-004, CHG-005, CHG-006

OPR-004, OPR-005, OPR-006

CRPR-003

6.11.12.3 Unsupported Requirements

OSR-046, OSR-050

6.12 RESTful Network APIs (OMA & GSMA)

6.12.1 Background

This clause is intended for analysing Network APIs defined in OMA (Open Mobile Alliance) and GSMA (Global System for Mobile Communications Association) that are used to open up service capabilities and assets in the Underlying Network to Applications. Although those APIs are provided either as RESTful style or SOAP Web Services style, this clause focuses on the RESTful style.

OMA and GSMA have defined standardized Network APIs for application developers to easily make use of existing mobile network capabilities such as messaging, location, payments, device capability discovery, call control, etc. for mobile application development. Many mobile operators already support some of these OMA/OneAPI standardized Network APIs in addition to proprietary APIs. As an option for service layer communicating with underlying network, oneM2M service layer can leverage these standardized APIs for the Mcn reference point to communicate with underlying networks for required services that networks provide.

The majority of the OMA Network APIs are the RESTful bindings of the existing Parlay X Web Service APIs. Additionally, OMA has defined and is still defining other required network APIs such as Customer Profile, Anonymous Customer Reference, Autho4API (i.e. usage of OAuth 2.0 in conjunction with the RESTful network APIs), Network Message Storage API, PushREST API, etc. based on the requirements obtained from another fora (e.g. GSMA OneAPI, RCS, etc.).

6.12.2 Status

6.12.2.1 Status of OMA RESTful Network APIs

This clause describes the brief description and status of the RESTful Network APIs defined by OMA. As mentioned previously, the most of OMA Network APIs are based on the existing Parlay X Web Service to provide the RESTful HTTP binding, but OMA has defined Network APIs by itself. The table 6.12.1 shows the brief description of each OMA RESTful Network API.

NOTE: The detail information (e.g. the relationship between OMA RESTful Network APIs and existing Parlay X APIs) each Network APIs can be found at "http://www.openmobilealliance.org/API/APIsInventory.aspx".

API Name	Rel	Description
File Transfer [i.56]	1.0	This specification introduces methods for a client to send files toward a server and to manage file transfer sessions.
Presence [i.57]	1.0	This specification introduces methods for a watcher (an application) to manage and retrieve presence information of a presentity.
Notification Channel [i.58]	1.0	This specification introduces methods for a client (e.g., a native application) to receive asynchronous notifications from a Notification Server about the events the client has subscribed to with one or more Enabler Servers.
Chat [i.59]	1.0	This specification introduces methods for a client chat application to send and receive 1-1 chat messages and manage the chat session.
Short Messaging [i.60]	1.0	This specification introduces methods for a client to send SMS messages to a terminal attached in the underlying network and to receive text messages. Additionally, checking delivery status and incoming messages is also included.
Third Party Call [i.61]	1.0	This specification introduces methods for a client to make and terminate a call session between calling participant and one or more called participant(s) and to obtain information regarding a call session and participant(s).
Address Book [i.62]	1.0	This specification introduces methods for a client to manage contacts and subscriptions to contact changes.
Messaging [i.63]	1.0	This specification introduces methods for a client to send MMS messages to a terminal attached in the underlying network and to receive text messages. Additionally, checking delivery status and incoming messages is also included.
Payment [i.64]	1.0	This specification introduces methods for a client to charge, refund, reserve and split amount to an end user's account and to retrieve payment transactions.
Device Capabilities [i.65]	1.0	This specification introduces methods for a client to retrieve device capabilities, such as device information and profiles, and push device configuration to a device.
Audio Call [i.66]	1.0	This specification introduces methods for a client to play audio and video messages to one or more call participants.
Call Notification [i.67]	1.0	This specification introduces methods for a client to manage subscriptions for call notifications, call direction notifications and media interaction notifications, and manage call event monitors.
Terminal Status [i.68]	1.0	This specification introduces methods for a client to retrieve the current device status information including accessibility, roaming, and connection type.

Image Share [i.69]	1.0	This specification introduces methods for a client to manage image share sessions and subscriptions to image share related event notifications.
Terminal Location [i.70]	1.0	This specification introduces methods for a client to obtain information about geographical location of a terminal.
Video Share [i.71]	1.0	This specification introduces methods for a client to manage video share sessions and subscriptions to video share related event notifications.
Customer Profile [i.72]	1.0	This specification introduces methods for a client to retrieve customer profile metadata (e.g., country, region, locality, area, and age).
ACR (Anonymous Customer Reference) [i.73]	1.0	This specification introduces methods for a client to create an ACR and query the status of an ACR. The ACR represents a unique identifier replacing a subscriber's secure information, such as MSISDN or phone number, ensuring privacy when interacting with web applications.
Capability Discovery [i.74]	1.0	This specification introduces methods for a client to retrieve own registered service capabilities and register/de-register service capabilities.
Converged Address Book [i.75]	1.0	This API allows applications to manage contacts in a contact collection, and members in member lists (e.g. an address list for presence or a group). The API also allows applications to refer to members in different member lists/groups from a contact. Applications can subscribe for notifications on changes in contacts and member lists. In addition, the API allows applications to enable a user to share contacts and member lists with other users, subject to user-defined authorization rules.
PushREST [i.76]	1.0	This specification defines the following operations. The Push Initiator (PI) is able to initiate the following operations to the Push Proxy Gateway (PPG): - Push Submission - Push Submission with Replace - Push Cancellation - Status Query - Client Capabilities Query The PPG is able to initiate the following message to the PI: - Result Notification
Network Message Storage [i.77]	1.0	Currently under development in OMA
Voice and Video over IP [i.78]	1.0	Currently under development in OMA Support of WebRTC voice and/or video calls.

Table 6.12.1: OMA RESTful Network APIs

6.12.2.2 Status of GSMA OneAPI

The GSMA OneAPI project is addressing deployment and operational considerations for 3rd party applications, and is re-using a subset of the Parlay X and OMA RESTful Network APIs for this. The table 6.12.2 shows the brief description of each GSMA OneAPI, and the details can be found at "http://www.gsma.com/oneapi/".

API Name	Rel	Relevant OMA RESTful Network APIs
ACR (Anonymous Customer Reference)	v4	RESTful Network API for Anonymous Customer Reference Management V 1.0
Customer Profile	v4	RESTful Network API for Customer Profile V 1.0
Data Connection Profile	v3	RESTful Network API for Terminal Status V 1.0
Device Capability	v3	RESTful Network API for Device Capabilities V 1.0
Payment	v3	RESTful Network API for Payment V 1.0
Location	v3	RESTful Network API for Terminal Location V 1.0
MMS	v3	RESTful Network API for Messaging V 1.0
SMS	v3	RESTful Network API for Short Messaging V 1.0
Voice Call Control	v3	RESTful Network API for Third Party Call V 1.0 RESTful Network API for Call Notification V 1.0 RESTful Network API for Audio Call V 1.0
Zonal Presence	v3 (Beta)	None NOTE: the relevant Network API is under discussion within OMA.

Table 6.12.2: GSMA OneAPI

6.12.4 Intended use

With those Underlying Networks (for example 3GPP and 3GPP2 networks) which support OMA Network APIs, network capabilities/services are exposed as resources on the northbound interface of networks. So the common service layer can make use of the APIs, perform defined operations on Mcn reference point for required services that networks support.

This clause describes several intended use cases for RESTful Network APIs as examples.

6.12.4.1 Location

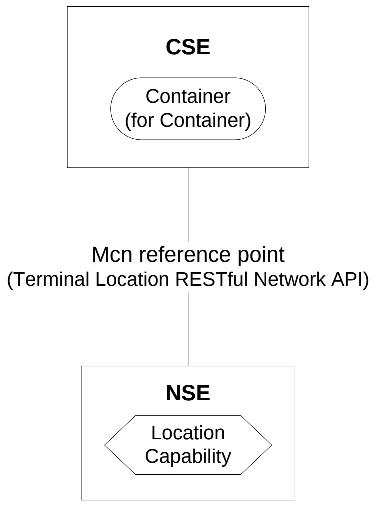


Figure 6.12: RESTful Network API Location (proposed)

With the underlying networks (for example 3GPP and 3GPP2) which support OMA Network APIs for terminal location or OMA Mobile Location Protocol, the user location is exposed as a resource over the Mcn reference point. When the location information of a target M2M Node needs to be stored in a container, the CSE can request the location using the RESTful Network API over the Mcn reference point. In this use case, the RESTful Network API for Terminal Location can be used and the CSE transforms the oneM2M configuration into the appropriate Network API configuration.

6.12.6 Key features

The brief features of each RESTful Network API are described in the clause 6.12.2

6.12.9 Security

Since RESTful Network APIs in OMA/GSMA OneAPI are technically identical to RESTful-HTTP protocol, these APIs are prone to the same vulnerabilities as standard web applications, including broken authentication, injection attacks, and cross-site scripting and cross-site request forgery.

Many HTTP security practices can be successfully applied for securing RESTful Network APIs.

6.12.10 Dependencies

RESTful Network APIs defined in OMA/GSMA OneAPI use HTTP as an application protocol for distributing state information and TCP/IP as a transport protocol to provide basic network connectivity.

6.12.11 Benefits

This clause lists the benefits of utilizing RESTful Network APIs defined in OMA and GSMA over Mcn reference point for supported services.

- Simple RESTful interface
- Requests and Responses of Network APIs do not require complex processing or validation
- For underlying network operators, lowers the barrier to entry and avoiding fragmentation
- For oneM2M service providers, reducing integration time and facilitating easy integration

6.12.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by the OMA and GSMA RESTful APIs is shown in the following table:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and the RESTful APIs comprise one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

Supported Requirements	Rationale	Related OMA RESTful NetAPI	Related GSMA OneAPI
OSR-006	The requirement states that oneM2M system shall be able to reuse the services offered by Underlying Networks by means of open access models.	ACR V 1.0 Customer Profile V 1.0 Terminal Status V 1.0 Device Capability V 1.0 Payment V 1.0	ACR Customer Profile Data Connection Profile Device Capability Payment

		Terminal Location V 1.0 Messaging V 1.0 Short Message V 1.0 Third Party Call V 1.0 Call Notification V 1.0 Audio Call V 1.0	Location MMS SMS Voice Call Control
OSR-047	The requirement states that oneM2M system shall report the geographical location information of M2M Devices/Gateways.	Terminal Location V 1.0	Location

Table 6.12.3: Supported requirements

6.13 ISA100.11a Protocol

The following clauses describe the ISA100.11a protocol.

6.13.1 Background

The ISA100 committee was formed in 2005 to define a family of industrial wireless automation standards. ISA100.11a [i.81] is the industrial wireless automation standard for process plants, and was officially released in 2009. ISA100 WG3 worked on this.

Release 1 of ISA100.11a addresses performance needs for control and monitoring applications where latencies on the order of 100 ms can be tolerated. Future releases to address critical and more delay sensitive applications such as emergency actions to ensure safety.

The ISA100 Wireless Compliance Institute (WCI) [i.80] functions as an operational group within The Automation Standards Compliance Institute (ASCI). As one of its activity, it certifies ISA100-compliant devices and systems. [i.83] It provides feedback to standard committees for improvement in the standards and has also specified certain application level profiles.

WCI members include the following: Agiliad, Apprion, Aramco Services Co, Armstrong International, Azbil, BP, Centero, Chevron, Control Data Systems, COSASCO, Crack Semiconductor, Eltav, ExxonMobil, Forbes Marshall, Flowserve, Fuji Electric, GasSecure, General Electric, Honeywell, New Cosmos Electric Co., Nexcom, Nivis, Pepperl+Fuchs, Perpetuum, R3 Sensors, Riken Keiki, Spirax Sarco, Scott Technologies, Shell Global Solutions, TLV Company Ltd., Yokogawa [i.82]

6.13.2 Status

ISA100.11a-2009 was approved by the ISA Standards and Practices Board in 2009.

6.13.3 Category and Architectural Style

ISA100.11a networks are IP-based multi-hop mesh networks, that are built using IEEE802.15.4 PHY and MAC layers. It also supports other topologies such as Star topology. As shown in Figure 6.13, ISA100.11a adds an upper data link layer and an application layer. This upper data link layer provides support for mesh routing, channel hopping and TDMA. These networks have built in mechanisms for time synchronization.

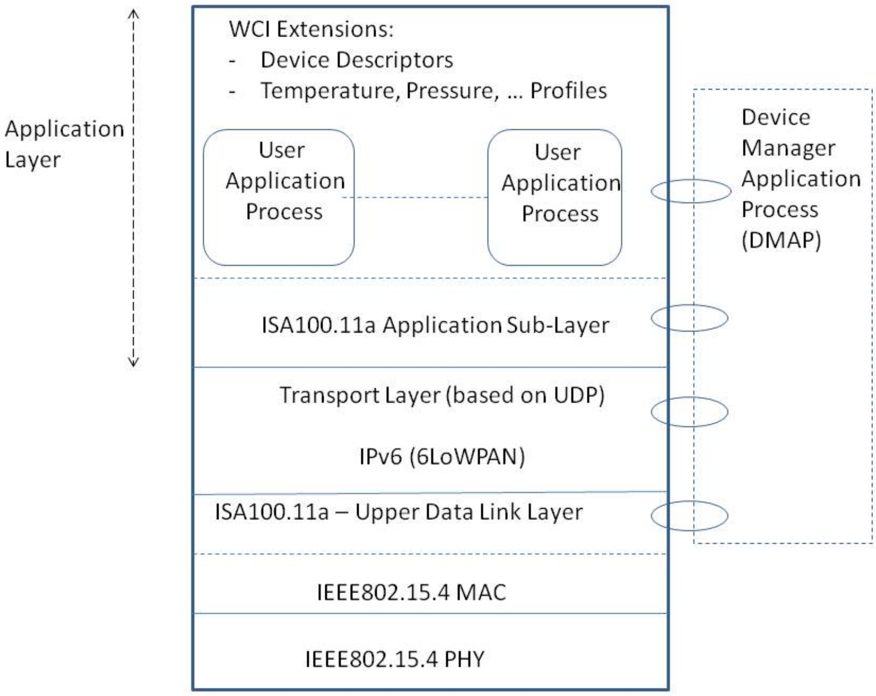


Figure 6.13: ISA100.11a Stack

The ISA100.11a application layer consists of the following:

- Upper Application Layer: Contains User Application Processes (UAPs) and Management Processes (MPs). Some UAPs are standardized and more can be added.
- Application Sub-Layer: Enables object oriented communication between peer objects in the same (or different) UAP(s). It also provides communication services to management processes.

Some standard objects specified by ISA100.11a include the following: 1 object per UAP for management purposes, device management object, UploadDownload object, Concentrator object, Alert reporting object, Alert receiving objects, and Gateway cache object. The Application layer of ISA100.11a allows tunnelling of non-ISA100.11a protocol packets and a Tunnel object is supported for this.

WCI specified ISA100 objects for temperature and pressure profiles. WCI uses device descriptor files to describe the capabilities and data structures of devices. A device descriptor file provides a description of each application object in that device. It helps the user to understand semantics of the data.

6.13.4 Intended use

ISA100 is designed for use within the process automation industry. Example applications include equipment monitoring (e.g. alerting, logging) and (closed / open loop) control

ISA100.11a based solutions can be deployed in industry areas such as: Oil & Gas, Mining, Chemicals, Life Sciences, Pulp & Paper, and Refining.

Performance expectations of ISA100.11a:

- Periodic monitoring where latency on the order of 100 ms can be tolerated
- High Reliability: 99.9%
- Sample intervals in few seconds
- 2-5 years battery life on end-devices

- Low data rate
- Range ~ 50 m
- Intended to cover large number of nodes

6.13.5 Deployment Trend

ISA100 products are available from companies such as: Gas Secure, GE Measurement and Control, Honeywell, Yokogawa, Apprion, Eltav, Cisco and Nivis. [i.83]

ISA100 success stories have been provided [i.84], and over 1 billion hours of operations for ISA100 devices around the world has been reported.

6.13.6 Key features

The following are some of the key features of ISA100.11a:

- ISA100.11a supports variety of mechanisms to ensure that data communication is reliable and secure.
- Each ISA100.11a device communicates at a pre-defined time and frequency. ISA100.11a stack supports variable length time slots with TDMA.
- Supports client-server model at application layer.
- ISA100.11a provides following basic services:
  - Publish / Subscribe service: Publishing is normally done via concentrator objects. A concentrator object collects various types of data from the device and packages that together for reporting. It supports assembly and disassembly of multiple values in the same ISA100.11a message.
  - Alert service where an alarm message is sent when some condition is satisfied (e.g. value of observed variable goes above a threshold) and another message is sent when this condition is cleared
- ISA100.11a tunnel capability at application layer allows carrying protocol messages of existing protocols such as HART, Modbus, Profibus and others.
- ISA100.11a supports establishment of a service contract before a device can start transmitting data.
- WCI has defined extensions for more complex applications.

6.13.7 Protocol Stack

ISA100 defines stacks for reliable and secure communication between field device and a gateway. It supports security mechanisms at MAC as well as transport layer.

ISA100.11a supports IPv6 addressing and uses IETF's IPv6 over Low rate Personal Area Network (6LoWPAN) standard and its transport layer is based on UDP. ISA100.11a application layer consists of Upper Application Layer and Application Sub-Layer.

6.13.8 Data Model

The native application layer of ISA100.11a supports reporting of analog data (such as pressure), binary data and block data (such as for waveform or firmware image).

Use of an Upload / Download object to communicate large amount of data (such as to transmit block of data representing a waveform from a field device to a gateway) is supported.

6.13.9 Security

Some security-related features of ISA100.11a are listed below:

- Supports message and device authentication, data confidentiality and data integrity.
- Hop-by-hop security is provided at MAC layer (using IEEE802.15.4 link layer methods)
- Message Integrity Code is computed at UDP layer to provide end-to-end message integrity at transport layer
- Supports AES-128 based encryption.
- Provides protection against relay attacks. Transport layer security uses time stamps in the nonce (for AES) that indicates when that packet was created. If packet is stale (e.g. if it was created more than T sec back), it is discarded.
- Use of symmetric keys is supported.
- Support of asymmetric key certificates is optional
- Supports dynamic key distribution using asymmetric keys based on public key cryptography. It enables over-the-air provisioning as well as automated re-keying.
- A Security Manager in the network manages and distributes keys.

6.13.10 Dependencies

The ISA100.11 stack is built using IEEE802.15.4 PHY/MAC and IETF 6LoWPAN.

6.13.11 Benefits

Benefits of ISA-100.11a include:

- Application layer is object oriented, flexible, modular and extensible.
- ISA100.11a application layer supports tunneling of other protocols. Thus, it allows data to be transferred via Modbus, Profibus, HART, Foundation Fieldbus or any other protocol.
- WCI object profiles are similar to that used by Foundation Fieldbus and that makes it easier to integrate these two types of systems.

6.13.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by ISA100 is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and ISA100 comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.13.12.1 Fully Supported Requirements

OSR-001, OSR-002, OSR-003, OSR-004, OSR-005, OSR-008, OSR-012, OSR-014, OSR-015, OSR-022, OSR-024, OSR-028, CRPR-001, OPR-001, SER-002, SER-003, SER-009, NFR-002.

6.13.12.2 Partially Supported Requirements

OSR-006, OSR-007, OSR-009, OSR-010, OSR-011, OSR-013, OSR-015, OSR-016, OSR-019, OSR-020, OSR-021, OSR-023, OSR-025, OSR-026, OSR-027,OSR-029, OSR-030, OSR-032, OSR-033, OSR-034, OSR-035, OSR-036, OSR-037, OSR-038, OSR-039, OSR-040, OSR-041, OSR-042, OSR-043, OSR-044, OSR-045, OSR-046, OSR-047, OSR-048, OSR-049, OSR-050, OSR-051, OSR-052, OSR-053, OSR-054, OSR-055, OSR-056, OSR-057, OSR-058, OSR-059, OSR-060, OSR-061, OSR-062, OSR-063, OSR-064, OSR-065, OSR-066, OSR-067, OSR-068, OSR-069, OSR-070, OSR-071, OSR-072, CRPR-002, CRPR-003, CRPR-004, CRPR-005

NOTE: ISA100.11a based protocols and systems can be enhanced to do variety of things that are not fully supported at present.

6.13.12.3 Unsupported Requirements

OSR-018
(This requirement is for cellular devices)

OPR-004
(ISA100.11a systems are specified for IEEE802.15.4-based devices, though theoretically one could consider supporting other interfaces as well)

SER-004 to SER-006
(These requirements are for UICC based devices. Some enhancements would be needed for support of UICC based devices with ISA100.11a systems)

6.14 WirelessHART® Protocol

The following clauses describe the WirelessHART® protocol.

6.14.1 Background

The HART Communication Foundation [i.86] was founded in 1993 and it is the technology owner and central authority on the HART protocol. The foundation has more than 250 members. Several process automation companies have been using HART based systems. HART7 [i.87], released in 2007, includes the wireless HART standard. It is compatible with existing (wired) HART devices and applications.

6.14.2 Status

As of April, 2010, WirelesHART is an approved standard by IEC, and is available as IEC 62591 [i.90]

6.14.3 Category and Architectural Style

Wireless HART supports mesh and star topologies and uses IEEE802.15.4 PHY / MAC layers. Its components include: Field device (sensor or actuators), Gateway, Network manager, Security manager, WirelessHART Handheld (to support direct access to adjacent field devices) and WirelessHART adapter (to connect existing HART devices to WirelessHART network). The Gateway is configured by the network manager and allows buffering of data, event notifications, command responses, and other messages.

As shown in Figure 6.14, WirelessHART has defined its own data link layer, network layer, transport layer and application layer. It uses channel hopping and TDMA-based slotted frames. The Network layer of WirelessHART provides routing and (end-to-end) security, and the Transport layer of WirelessHART supports acknowledged as well as un-acknowledged communication.

WirelessHART uses the command-based application layer as used in the HART systems. In addition to the command types supported within HART systems, WirelessHART supports Wireless command types. WirelessHART‘s application layer protocol operates in the host / slave mode, and supports data publishing.

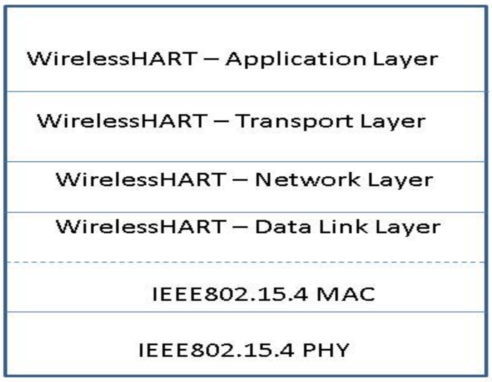


Figure 6.14: WirelessHART® stack

6.14.4 Intended use

WirelessHART is designed for use within process automation industry. Example applications include equipment monitoring and (closed / open loop) control

6.14.5 Deployment Trend

According to industry sources, as reported by the HART Communication Foundation [i.86], there are:

- more than 30 million HART devices are installed in the process automation industry worldwide,
- 75% of process measurement and control devices installed worldwide using HART communication.

6.14.6 Key features

The following are some of the key features of WirelessHART:

- Reliable and secure protocol
- Supports up to 8 process variables in a single message
- Supports Time stamped data
- Supports transfer of large data streams (such as radar level curves)

6.14.7 Protocol Stack

The WirelessHART protocol stack is shown above in Figure 6.14. It extends the HART protocol at application layer with wireless command type and defines its own transport, network and data link layers.

6.14.8 Data Model

WirelessHART uses Device Descriptor (DD) files where a DD file describes features and functions of a device. A DD is created as a text file and then converted into a standard binary file. This DD is written in conformance with a Device

Description Language. The HART Communication Foundation manages a library of Manufacturers Device Descriptions. Some of the supported data types include fix and floating point numbers, bit and byte arrays, enumerations, time and text.

6.14.9 Security

Some security related features of WirelessHART are listed below:

- Provides hop-by-hop security (at layer 2) and end-to-end security (at network layer)
- Supports use of symmetric keys
- Uses AES-128 block cipher for encryption and message authentication
- Layer 2 provides hop-by-hop security and uses 32-bit Message Integrity Check for each frame
- Supports separate Join key per-device. This is used to authenticate the joining device with network manager

6.14.10 Dependencies

The WirelessHART stack is built using IEEE802.15.4 PHY/MAC and uses its own data link, network and transport layer. It also uses HART command types at application layer to stay compatible with existing (wired) HART deployments.

6.14.11 Benefits and Constraints

6.14.11.1 Benefits

Benefits of WirelessHART include:

- Compatible with wired HART systems
- Uses Device Description Language to describe service and confgiuration of field devices.
- Uses only HART at application layer and potentially keeps it simple for process automation industry *(also in constraints below)*
- Extensive installed base

6.14.11.2 Constraints

Constraints of WirelessHART include:

- Key management is not supported.
  (This is a limitation also shared with other IEEE802.15.4 based systems. IEEE802.15.9 is working to standardize key management.)
- Supports only HART at application layer
- Does not use all-IP stacks
  (It is theoretically possible to change network and transport layer of WirelessHART to IP stacks supported by IETF)

6.14.12 Support of oneM2M requirements

Support of oneM2M Requirements [i.2] by WirelessHART is shown in the following clauses:

NOTE: Many requirements from TS-0002 depend on the architecture of overall M2M system and WirelessHART comprises one aspect of this system. To specifically compare with each requirement, one would need to take several assumptions about system components, and compliance would vary depending on behaviour of those other components. Thus, only a subset of the requirements are highlighted here.

6.14.12.1 Fully Supported Requirements

OSR-002, OSR-003, OSR-004, OSR-005, OSR-008, OSR-010, OSR-011, OSR-012, OSR-014, OSR-015, OSR-022, OSR-024, OSR-028, CRPR-001, OPR-001, SER-002, SER-003, SER-009, NFR-002.

6.14.12.2 Partially Supported Requirements

OSR-006, OSR-007, OSR-009, OSR-013, OSR-016, OSR-019, OSR-020, OSR-021, OSR-023, OSR-025, OSR-026, OSR-027,OSR-029, OSR-030, OSR-032, OSR-033, OSR-034, OSR-035, OSR-036, OSR-037, OSR-038, OSR-039, OSR-040, OSR-041, OSR-042, OSR-043, OSR-044, OSR-045, OSR-046, OSR-047, OSR-048, OSR-049, OSR-050, OSR-051, OSR-052, OSR-053, OSR-054, OSR-055, OSR-056, OSR-057, OSR-058, OSR-059, OSR-060, OSR-061, OSR-062, OSR-063, OSR-064, OSR-065, OSR-066, OSR-067, OSR-068, OSR-069, OSR-070, OSR-071, OSR-072, CRPR-002, CRPR-003, CRPR-004, CRPR-005

NOTE: It is possible to enhance WirelessHART protocols to do variety of things that are not fully supported at present.

6.14.12.3 Unsupported Requirements

OSR-001
(WirelessHART defines its own network and transport layers. Theoretically, these could be replaced with IP based protocols.)

OSR-018
(This requirement is for cellular devices)

OPR-004
(WirelessHART systems are specified for IEEE802.15.4-based devices though theoretically one could consider supporting other interfaces as well.)

SER-004 to SER-006
(These requirements are for UICC based devices. Some enhancements would be needed for support of UICC based devices by WirelessHART systems).

7 Summary

The following tables summarize how selected traits are addressed by the analysed protocols.

Note: the following tables compare M2M application Layer Protocols. Other IoT & M2M protocols are listed in Clause 6 - Analysis of Protocols, including the areas of capillary networks, application profiles & network server to application interfaces.

Protocol / Traits	Architecture Style	Intended or Actual Deployment	Relative position to other protocols	Data Model / Data Representation	Messaging (only a subset of features indicated here)	Security (only a subset of mechanisms listed here)
CoAP	Client / server model. P2P. RESTful	IoT networks with low power constrained sensors such as smart metering	Above UDP (or DTLS/ UDP)	Plain text, XML, JSON, EXI, octet-stream… Support for content negotiation	Request / Response, Pub-Sub	Largely depends on lower layers (DTLS...)

Protocol / Traits	Architecture Style	Intended or Actual Deployment	Relative position to other protocols	Data Model / Data Representation	Messaging (only a subset of features indicated here)	Security (only a subset of mechanisms listed here)
MQTT	Client / Server model. Brokered style.	Low bandwidth, high latency networks (e.g. HealthCare, Energy)	MQTT runs above TCP (or TLS/TCP). (MQTT-SN runs over UDP)	No formal data model	Publish-Subscribe	Authentication: Userid / password can be passed in a packet. SSL / TLS can be used.
HTTP as RESTful API	RESTful	WWW	Above TCP (or TLS/TCP)	XML, JSON, etc. Support for Content negotiation	Request - Response	Largely depends on lower layers (SSL / TLS…)
XMPP	Availability for Concurrent Transactions (ACT) style for carrying out asynchronous end-to-end exchange of structured data	Instant Messaging and Presence Applications , Jabber	Above TCP (or TLS/TCP)	XML, EXI	Publish-Subscribe	SASL, TLS, lower layer security
WebSockets	Full duplex communication over TCP	Low latency, high performance web applications	Above TCP. Uses HTTP for initial handshake	JSON, etc.	Full duplex communication same over TCP socket.	TLS
DDS	Data centric model. (Virtual) Global Data space and broker-less Peer-to-Peer model	Several segments such as health care, UAVs, asset tracking, etc.	Can run over UDP, TCP, shared memory and other transport types	DSSI defines a standard data format based on extension of Common Data Representation. Named topics, user defined data types	Real-time Publish-Subscribe	TLS and some OMG specific security methods. Vendor specific extensions also available.
Modbus	Message passing. Client-Server model.	Process Automation Industry, Power substation applications	Over serial interfaces (RS-232 / 485), over TCP/IP/ Ethernet…	Bit-addressable and 16-bit word addressable…	Request / Response mode	Depends on other security mechanisms.
DNP3	Client-Server model.	Electric (Power System) and water utility companies	Over serial interfaces (RS-232 / 485) and over TCP/IP/ Ethernet	Binary input / output, Analog input / output, Counter input	Request-Response model, Report-by-exception.	Mutual authentication using challenge response mechanisms

Protocol / Traits	Architecture Style	Intended or Actual Deployment	Relative position to other protocols	Data Model / Data Representation	Messaging (only a subset of features indicated here)	Security (only a subset of mechanisms listed here)
UPnP Cloud	Client-Server model	Home Automation	Above TCP (such as over XMPP/TCP or over HTTP (extensions) / TCP)	State variables of complex type expressed in XML.	Publish Subscribe	Uses SASL for authentication, TLS
ISA100.11a	Client Server model for ISA100.11a application layer	Process automation	Layer 2+ stacks over IEEE 802.15.4 type of networks	Analog, binary, block data (such as for waveform and firmware image)	Publish – Subscribe, Alert	Authentication, Confidentiality, Integrity
Wireless HART	Master-Slave mode for IEEE802.15.4 based networks	Process automation	Layer 2+ stacks over IEEE 802.15.4 type of mesh / star networks	Fix and floating point numbers, bit and byte arrays, enumerations, time and text.	Command based application layer used in HART systems. Master-Slave mode and data publishing	Hop-by-hop (Layer 2) and end-to-end (network) layer security

Table 7.1: M2M Application Layer Protocols – Summary (I of II)

Protocol / Traits	Synchronization mechanisms	QoS	Discovery	Multicast	SDOs
CoAP	No internal mechanism. (Could use NTP, IEEE 1588v2 and other mechanisms.)	Confirmable or non-confirmable message modes	Support for discovery. Uses concept of resource directory	Supports IP multicast	IETF
MQTT	Relies on external mechanisms	Three assurance levels for message delivery (deliver message at most once, exactly once, at least once)	No automatic discovery.	-- (Depends on external protocols)	OASIS
HTTP as RESTful API	No	Reliability via TCP	No	--	It is an architecture style. Protocol part: W3C and IETF.
XMPP	No	Reliability over TCP	Service discovery	Syntax for sending messages to multiple recipients is supported	IETF, XEP
WebSockets	No	Reliability over TCP	No	--	IETF, W3C

DDS	Relies on external mechanisms	20+ QoS policies in terms of latency budget, reliability etc. Reliability provided by DDSI protocol.	Automatic discovery of publishers and subscribers	If DDS over UDP, one could use IP multicast	OMG
Modbus	For Modbus / TCP, NTP or some other protocol can be used. For Modbus over serial interfaces, mechanisms can be built on top of Modbus protocol.	TCP can provide reliability for Modbus-TCP.	No	No	Modbus.org
DNP3	In-built time synchronization mechanism as part of DNP3 standard	Reliable data transfer through the use of time-stamping	No	If using over UDP, one could use IP multicast	IEEE, DNP user group
UPnP Cloud	DCP level (like synchronized video playout)	Reliability via TCP or other protocols	Yes	Multicast events mapped to XMPP Pub-Sub	UPnP Forum
ISA100.11a	ISA networks need time synchronization mechanisms	Can be supported by different layers of the stack	Neighbour discovery	Depends on other layers of the stack	ISA, Uses components such as 6LoWPAN from IETF
Wireless HART	Time Synchronization needed	Can be supported	Neighbour discovery	Depends on other layers of the stack	IEC

Table 7.2: M2M Application Layer Protocols – Summary (II of II)

Performance: Following should be noted:

- Some of these protocols (and associated systems) such as Modbus, DNP3, Wireless HART and ISA100.11a are optimized for industrial applications.
- CoAP and MQTT-SN are simple and efficient protocols for IoT applications. Though MQTT runs over TCP, MQTT-SN (MQTT for Sensor Networks) runs over UDP.
- DDS offers good capabilities for the scenarios when there are several applications running on a node.

The following text is to be used when appropriate:

Proforma copyright release text block

This text box shall immediately follow after the heading of an element (i.e. clause or annex) containing a proforma or template which is intended to be copied by the user. Such an element shall always start on a new page.

Notwithstanding the provisions of the copyright clause related to the text of the present document, OneM2M grants that users of the present document may freely reproduce the <proformatype> proforma in this {clause|annex} so that it can be used for its intended purposes and may further publish the completed <proformatype>.

Annex A
List of M2M-related Protocols (Informative)

NOTE: The following list table has been created for reference from publicly-available sources, and no representation is made regarding the accuracy or timeliness of the information it contains. In addition, the appearance or omission of any M2M-related information in this list does not imply either the intention, or lack of intention, to undertake any normative or other work within oneM2M.

Table A.1: M2M-related Protocols

Short Name (docs link)	Full Name / Description	Originating org (source link)	Notes
1-Wire®	1-Wire®	[Dallas-Maxim]	
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks	IETF 6LoWPAN	RFC 4944
AllJoyn™	AllJoyn™	AllJoyn Alliance [Qualcomm]	(open source)
ANSI	~~	ANSI	
ANSI C12.18	Protocol Specification for ANSI Type 2 Optical Port	ANSI / NEMA	
ANSI C12-21	Protocol Specification for Telephone Modem Communication	ANSI / NEMA	
ANSI C12.22	Interfacing to Data Communication Networks	ANSI / NEMA	Advanced Metering Infrastructure (AMI)
ANT+	ANT+	ThisIsAnt [Dynastream Inc]	
BACnet™	Building Automation & Control Network	ASHRAE SSPC 135 BACnet International	
BâtiBUS,	*Bâtiment-Bus -superceded-*		*see KNX*
BitXML	BitXchange Markup Language	BitXML [Your Voice S.P.A.]	
Bluetooth	~~	Bluetooth SIG	IEEE 802.15.1
Bluetooth HDP	Bluetooth Health Device Profile	Continua Health Alliance Bluetooth SIG	**Partner Type 2**
Bluetooth LE / SMART	Bluetooth Low Energy / Smart Devices	Bluetooth SIG	
C-Bus	C-Bus	[Clipsal / Schneider Electric]	
CANOpen	Controller Area Network - Open	CANOpen Forum [CiA. e.V.]	EN 50325-4 2002 Part 4
CC-Link	CC-Link	CC-Link Partner Association	SEMI E54.12-0701E

		[Mitsubishi]	
CEBus	Consumer Electronics Bus	CEA (was EIA)	EIA 600
CIP	Common Industrial Protocol	Open DeviceNet Vendors Association [Rockwell Automation]	
CoAP	Constrained Application Protocol	IETF CORE WG	**See Clause 6.1**
CompoNet	CompoNet (CIP) on TDMA Technology	Open DeviceNet Vendors Association	
Contiki	Contiki Operating System	Contiki Project	(open source)
ControlNet	ControlNet (CIP) on CTDMA Technology	Open DeviceNet Vendors Association	
DALI	Digital Addressable Lighting Interface	DALI	IEC 62386
DLMS	*Device Language Message Specification*		*see IEC 62056*
DASH7	(ISO 18000) - Dash 7	DASH7 Alliance	ISO/IEC 18000-7
DDS	Data Distribution Service for Real-Time Systems	OMG	
DDS-RTPS	DDS Real-Time Publish-Subscribe	OMG	
DECT™ ULE	Digital Enhanced Cordless Telecommunications - Ultra Low Energy	ETSI TC DECT	TS 102 939-1
DeviceNet	DeviceNet (CIP) on CAN Technology	Open DeviceNet Vendors Association [Allen-Bradley / Rockwell]	
DNP	Distributed Network Protocol	IEEE / DNP	IEEE Std 1815™
Dynet 1 / 2	Dynalite Network	[Philips Dynalite]	RS-485
E5	(Ease, Energy, Efficiency, Environment and Earth) Smart Thermostat	[EarthNetworks]	
Eclipse	~~	Eclipse Foundation	
Concierge (proposed)	Lightweight, embeddable OSGi framework	Eclipse Foundation	(open source)
Kura (proposed)	Java M2M framwork	Eclipse Foundation	(open source)
Lua API	Scripting API	Eclipse Foundation	(open source)
Mihini / M3DA	Mihini/M3DA Specification	Eclipse Foundation	(open source)
Paho	Implementations of Open and Standard Messaging Protocols	Eclipse Foundation	(open source)
Ponte (proposed)	M2M to REST bridge	Eclipse Foundation	(open source)
SCADA	open Supervisory Control and Data Acquisition	Eclipse Foundation (was openSCADA)	(open source)
E-DCP	Ericsson Device Connection	[Ericsson]	

	Platform		
EHS	*European Home Systems -superceded-*		*see KNX*
EIB	*European Installation Bus -superceded-*		*see KNX*
Energyhub	Mercury™ smart thermostat platform	[Energyhub]	
EnOcean	EnOcean Weqipment Profiles (EEP)	EnOcean Alliance [EnOcean / Siemens]	EN 50090, ISO/IEC 14543
ETSI M2M TS 102 921	Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces	ETSI M2M	**Partner Type 1 oneM2M Pool Document**
EXALTED	EXpAnding LTE for Devices	Exalted consortium [Eurescom]	EC FP7, 2010-2013
FieldBus	FieldBus	Fieldbus Foundation	IEC 61158
FlatMesh	Remote Condition Monitoring	[Senceive]	IEEE 802.15.4
flexWARE	Flexible Wireless Automation in Real-Time Environments	flexWARE Interest Group (FIG)	EU FP7
HBS	Honeywell Building Solutions	[Honeywell]	
IEC	~~	IEC	
IEC 60870-5-xxx	Telecontrol (Supervisory Control and Data Acquisition)	IEC TC57 WG03	IEC 101 IEC 103 IEC 104
IEC 61107	Smart Meter Communications Protocol	IEC TC57	
IEC 61850	Electrical Substation Automation.	IEC TC57	
IEC 62056 DLMS/ COSEM	Device Language Message Specification/Companion Specification for Energy Metering	IEC TC13 WG 14 DLMS User Association	
IEC 62351	Security	IEC TC57 WG15	
INSTEON	Dual-mesh (RF/PL) Home Management Metwork	Insteon [SmartLabs, Inc.]	
IEEE	~~	IEEE	
1451.x	Smart Transducer Interface for Sensors and Actuators	IEEE	
P1451.1.4	Smart Transducer Interface for Sensors, Actuators, and Devices - XMPP	IEEE IM/ST - TC9	ISO/IEC/IEEE 21451-1-4
802.11a **802.11b** **802.11g** **802.11n**	Wi-Fi	IEEE Wi-Fi Alliance	
802.11p	WAVE - Wireless Access in Vehicular Environments	IEEE	IEEE 1609
802.11ag	WiGig	IEEE Wi-Fi Alliance	

		(was WiGig)	
802.11ah (in progress)	Sub 1 GHz (S1G) Wireless Sensor Network for Smart Metering	IEEE	
802.15	Wireless Personal Area Network (WPAN)	IEEE	
802.16	WiMAX Wireless Metropolitan Area Networks	WiMAX Forum IEEE	
IETF	~~	IETF	
IP (v4)	Internet Protocol	IETF	RFC791 Updated by: RFC1349, RFC2474, RFC6864
IPv6	Internet Protocol version 6	IETF	RFC2460 Updated by: RFC5095, RFC5722, RFC5871, RFC6437, RFC6564, RFC6935, RFC6946
TCP	Transmission Control Protocol	IETF	RFC793 Updated by: RFC1122, RFC3168, RFC6093, RFC6528
UDP	User Datagram Protocol	IETF	RFC768
Instabus	*-superceded-*		*see KNX*
IPDR	IP Data Record	TM Forum	
IrDA	~	Infrared Data Association	
FIR	Fast IrDA	Infrared Data Association	
SIR	Serial Infrared	Infrared Data Association	
IRsimple™	IrDA Simple (high-speed wireless)	Infrared Data Association	
ISA100.11a	ISA100.11a	ISA ISA100 Wireless Compliance Institute (WCI)	
ISO 21215 CALM M5	Communication Access for Land Mobile	ISO TC 204/WG 16	
KNX	KNX (Konnex)	KNX Association	ISO/IEC 14543-3 CENELEC EN

			50090 CEN EN 13321-1 [CN] GB/Z 20965
HGI (in progress)	~~	Home Gateway Initiative (HGI)	**Partner Type 2**
RWD036	Smart Home Architecture and System Requirements	Home Gateway Initiative (HGI)	
RWD043	Requirements for RP1 on the Smart Home Platform	Home Gateway Initiative (HGI)	
GWD042	Smart Home Appliance (Device) Model Template	Home Gateway Initiative (HGI)	
HL7	Health Level Seven	Health Level Seven International	
Jabber	-		*see XMPP*
LonWorks®	Control Network Protocol Specification	LonMark [Echelon]	ANSI/CEA-709.1-B SO/IEC 14908-1)
M2MXML	Machine-To-Machine XML-based Protocol	M2MXML Project	(open source)
Mango	Mango Automation	[Serotonin Software]	(open source)
M-Bus	Meter Bus	M-Bus Usergroup	EN 13757-2, -3
MiWi	Microchip P2P Wireless Protocol	[Microchip Tech]	IEEE 802.15.4
Modbus	Modicon Bus	Modbus Organization [Schneider Automation] (was Modicon)	
Modbus TCP (in progress)	Modicon Bus TCP/IP	Modbus Organization [Schneider Automation]	
MyriaNed®	Self organizing Wireless Sensor Network	[DevLab] [Chess]	
OASIS	~~	OASIS	
AMQP	Advanced Message Queuing Protocol	OASIS AMQP TC [JPMorgan-Chase]	
MQTT	Message Queuing Telemetry Transport	OASIS MQTT TC MQTT.org [IBM/Eurotec (Arcom)]	**See Clause 6.2**
oBIX	Open Building Information Exchange	OASIS oBIX TC oBix	
OMA M2M Enablers	~~	Open Mobile Alliance	**Partner Type 2**
CPNS	Converged Personal Network Services	Open Mobile Alliance	
DM 1.3	Device Management	Open Mobile Alliance	
DM 2.0	Device Management	Open Mobile Alliance	
GwMO	Gateway Management Object	Open Mobile Alliance	
LWM2M	Lightweight M2M protocol	Open Mobile Alliance	
M2M DC	M2M Device Classification	Open Mobile Alliance	
OCMAPI	Open Connection Manager API	Open Mobile Alliance	
oneNet	Low Power Wireless Protocol	ONE-NET	(open source)

OpenSCADA	*-superceded-*	-	*see Eclipse SCADA*
OpenTag	*-superceded-*	-	*see DASH7*
OpenWSN	Open Wireless Sensor Networks	OpenWSN Project (was UC Berkeley)	(open source)
OSGi™ R5	OSGi R5 for embedded devices	OSGi™ Alliance	
OSGP	Open Smart Grid Protocol	ETSI (was ISG OSG)	GS OSG 001 Ver. 1.1.1 with ISO/IEC 14908
OSIAN	Open Source IPv6 Automation Network	OSIAN Project	(open source)
Profibus	Process Field Bus	Profibus & Profinet International (PI)	
RFID	Radio-Frequency IDentification		
EN 300 220	ERM Short Range Devices	ETSI	
EN 302 208	ERM Radio Frequency Identification Equipment	ETSI	
EPC Gen2	EPCglobal UHF Class 1 Generation 2	EPCglobal	
ISO/IEC 14443	HighFID Proximity Card	ISO/IEC	
ISO/IEC 15693	HighFID non-contact Smart Tags	ISO/IEC	
ISO/IEC 18000	Radio frequency identification for item management	ISO/IEC	
ISO/IEC 18092	Near Field Communication NFCP-1	ISO/IEC	
ISO/IEC 21481	Near Field Communication NFCP-2	ISO/IEC	
RS-232	*-superceded-*	*EIA*	*see TIA-232-F*
RS-422	*-superceded-*	*EIA*	*see TIA -422*
RS-485	*-superceded-*	*EIA*	*see TIA-485*
RuBee	High security Wireless Asset Visibility Network	RuBee [Visible Assets]	IEEE 1902.1, 1901.2
Sinec H1	Siemens Ethernet Control - H1 *-legacy-*	[Siemens]	
SOAP	Simple Object Access Protocol	W3C	
SMART	*-superceded-*		*see Bluetooth SMART*
SmartBus	SmartBus Home Automation Control (HAC)	Smart Home Group [Digitcom Technology]	
SMS	Short Message Service	3GPP	TS 23.040
SNMP	~~	IETF OPSA WG (was SNMP WG)	
SNMP v3	Simple Network Management Protocol v3	IETF OPSA WG	RFC3411
SNMP MIB	Management Information Block	IETF OPSA WG	RFC3418
TIA	~~	TIA	**Partner Type 1**
TIA-232-F	Interface Between Data	TIA TR-30	

	Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange	(was EIA RS-232)	
TIA-422	Electrical Characteristics of the Balanced Voltage Digital Interface Circuit	TIA TR-30 (was EIA RS-422)	
TIA-485	Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems	TIA TR-30 (was EIA RS-485)	
TIA-4940.020	Smart Device Communications Protocol Aspects	TIA TR-50	**Partner Type 1 oneM2M Pool Document**
UPnP	Universal Plug and Play	UPnP Forum	
TR-069	CPE WAN Management Protocol	Broadband Forum (BBF)	**Partner Type 2**
VSCP	Very Simple Control Protocol	VSCP Project [Grodans Paradis AB]	(open source)
WAP	Wireless Application Protocol	Open Mobile Alliance (was: WAP Forum)	
WAVE2M	Open Low-Power Wireless	WAVE2M	
Wavenis	*-superceded-*		*see WAVE2M*
Weightless 1.0	Low-power White-space Wireless Network	Weightless SIG [Neul]	
Wibree	*-superceded-*		*see Bluetooth SMART*
WirelessHART ®	Wireless Highway Addressable Remote Transducer	HART Communication Foundation	IEEE 802.15.4 / IEC 62591
Wireless HD	Wireless High-Definition Digital Interface	WirelessHD Consortium	
Wireless USB	Wireless Universal Serial Bus	USB Implementers Forum	
WorldFIP	World Factory Instrumentation Protocol	WorldFIP	
X-10	X-10	[X-10 (USA)]	
xAP	XAP Home Automation Protocol	XAP Forum	(open source)
xPL	xPL Home Automation Project	xPL Project	
XMPP	eXtensible Messaging and Presence Protocol	IETF XMPP WG	RFC6120
XMPP XEP	eXtensible Messaging and Presence Protocol Extensions	XMPP Standards Foundation	
ZigBee®	ZigBee 2012	ZigBee Alliance	IEEE 802.15.4
ZigBee IP	ZigBee IPv6 for Smart Energy	ZigBee Alliance	
ZigBee RF4CE	ZigBee for Consumer Electronics	ZigBee Alliance	
SEP 2	Smart Energy Profile 2.0	CSEP	

		ZigBee Alliance	
Z-Wave	Wireless RF-based Communications Technology	Z-Wave Alliance Z-Wave	ITU G.9959

Annex B
Definitions of Radio metrics for Technologies used for M2M releated Protocols (Informative)

NOTE: The following definitions are quoted for reference from publicly-available sources, and no representation is made regarding the accuracy or timeliness of the information it contains. In addition, the appearance or omission of any M2M-related information in this list does not imply either the intention, or lack of intention, to undertake any normative or other work within oneM2M.

		ZigBee	**Bluetooth**	**802.11b**	**802.11g**	**802.11a**	**802.11n**	**UWB**
Throughput	**Mbps**	0.03	1-3	11	54	54	200	200
Max range	**ft**	75	30	200	200	150	150	30
Sweet spot	**Mbps-ft**	.03@75	1-3@10	2@200	2@200	36@100	100@100	200@10
Service	**bps-ft2**	530	314M	251G	251G	1.13T	3.14T	62G
Power	**mW**	30	100	750	1000	1500	2000	400
Bandwidth	**MHz**	0.6	1	22	20	20	40	500
Spectral efficiency	**b/Hz**	0.05	1	0.5	2.7	2.7	5	0.4
Power efficiency1	**mW/Mbps**	1000	100	68	19	27	10	2
Power efficiency2	**mAh/GB**	2211	67	46	12	18	7	1.3
TTGB	**Time**	3.1 day	2.2 hr	12 min	2.5 min	2.5 min	40 sec	40 sec
Price	**US$**	$2	$3	$5	$9	$12	$20	$7
Note: TTGB: Time To Generate (the time to live for a security session key).								

Table B-1: Wireless access technologies in the M2M landscape

B.1 Bluetooth® Wireless Technology

- Bluetooth wireless technology is geared towards voice and data applications
- Bluetooth wireless technology operates in the unlicensed 2.4 GHz spectrum
- The range of Bluetooth wireless technology is application specific. The Bluetooth Specification mandates operation over a minimum distance of 10 meters or 100 meters depending on the Bluetooth device class, but there is not a range limit for the technology. Manufacturers may tune their implementations to support the distance required by the use case they are enabling.
- The peak data rate with EDR is 3 Mbps
- Bluetooth wireless technology is able to penetrate solid objects
- Bluetooth technology is omni-directional and does not require line-of-sight positioning of connected devices
- Security has always been and continues to be a priority in the development of the Bluetooth specification. The Bluetooth specification allows for three modes of security

B.2 ZigBee (IEEE 802.15.4)

The promoter companies of the ZigBee Alliance include: Philips, Honeywell, Mitsubishi Electric, Motorola, Samsung, BM Group, Chipcon, Freescale and Ember; more than 70 members

- Capacity of 250 Kbits at 2.4 GHz, 40 Kpbs at 915 Mhz, and 20 Kpbs at 868 Mhz with a range of 10-100 M
- Its purpose is to become a wireless standard for remote control in the industrial field
- The ZigBee technology is targeting the control applications industry, which does not require high data rates, but must have low power, low cost and ease of use (remote controls, home automation, etc.)
- The specification was formally adopted in December 2004
- Security was not considered in the initial development of the specification. Currently there are three levels of security

B.3 Ultra-Wideband (UWB)

- UWB technology for Personal Area Networks offers a unique combination of low power consumption (~1mW/Mbps) and high data throughput (up to 480 Mbps).
- WiMedia UWB is an internationally recognized standard (ECMA-368, ISO/IEC 26970 and ECMA-369, ISO/IEC 26908) and has regulatory approval in major markets worldwide, including US, EU, Korea and Japan. Additional regions, e.g. China and Canada are expecting regulatory approval in the near future.
- Ideally, it will have low power consumption, low price, high speed, use a wide swath of radio spectrum, carry signals through obstacles (doors, etc.) and apply to a wide range of applications (defense, industry, home, etc.)
- WiMedia UWB takes a "Common Radio Platform" approach allowing the same radio to be used for a variety of applications.
- WiMedia UWB allows for data rates up to 480Mbps at ranges of several meters and a data rate of approximately 110 Mbps at a range of up to 10 meters
- While Wireless USB has initially utilized UWB technology, it is expected that the Bluetooth high speed solution will not suffer the same performance and interoperability issues due to the specification development and qualification process employed by the Bluetooth SIG.
- The Bluetooth SIG announced in May 2005 its intentions to work with UWB to develop a high rate Bluetooth specification on the UWB radio

B.4 Certified Wireless USB

- Speed: Wireless USB is projected to be 480 Mbps up to 2 meters and 110 Mbps for up to 10 meters. Wireless USB hub can host up to 127 wireless USB devices
- Wireless USB will be based on and run over the UWB radio promoted by the WiMedia Alliance.
- Allows point-to-point connectivity between devices and the Wireless USB hub
- Intel established the Wireless USB Promoter Group in February 2004
- The USB Implementers Forum, Inc. (USB-IF) tests and certifies the "certified Wireless USB" based wireless equipment

B.5 Wi-Fi (IEEE 802.11)

- Bluetooth technology uses a fifth of the power of Wi-Fi
- The Wi-Fi Alliance tests and certifies 802.11 based wireless equipment

- 802.11a: This uses OFDM, operates in the 5 GHz range, and has a maximum data rate of 54 Mbps
- 802.11b: Operates in the 2.4 GHz range, has a maximum data rate of 11 Mbps and uses DSSS. 802.11b is the original Wi-Fi standard
- 802.11g: Operates in the 2.4 GHz range, uses OFDM and has a maximum data rate of 54 Mbps. This is backwards compatible with 802.11b
- 802.11e: This standard will improve quality of service
- 802.11h: This standard is a supplement to 802.11a in Europe and will provide spectrum and power control management. Under this standard, dynamic frequency selection (FS) and transmit power control (TPC) are added to the 802.11a specification
- 802.11i: This standard is for enhanced security. It includes the advanced encryption standard (AES). This standard is not completely backwards compatible and some users will have to upgrade their hardware. The full 802.11i support is also referred to as WPA2
- 802.11k: Under development, this amendment to the standard should allow for increased radio resource management on 802.11 networks
- 802.11n: This standard is expected to operate in the 5 GHz range and offer a maximum data rate of over 100 Mbps (though some proposals are seeking upwards of 500 Mbps). 802.11n will handle wireless multimedia applications better than the other 802.11 standards
- 802.11p: This standard will operate in the automotive-allocated 5.9 GHz spectrum. It will be the basis for the dedicated short range communications (DSRC) in North America. The DSRC will allow vehicle to vehicle and vehicle to roadside infrastructure communication
- 802.11r: This amendment to the standard will improve users' ability to roam between access points or base stations. The task group developing this form in spring/summer 2004
- 802.11s: Under development, this amendment to the standard will allow for mesh networking on 802.11 networks. The task group developing this formed in spring/summer 2004.

B.6 Radio Frequency Identification (RFID)

There are over 140 different ISO standards for RFID for a broad range of applications

- With RFID, a passive or unpowered tag can be powered at a distance by a reader device. The receiver, which must be within a few feet, pulls information off the 'tag,' and then looks up more information from a database. Alternatively, some tags are self-powered, 'active' tags that can be read from a greater distance
- RFID can operate in low frequency (less than 100 MHz), high frequency (more than 100 MHz), and UHF (868 to 954 MHz)
- Uses include tracking inventory both in shipment and on retail shelves

B.7 Near Field Communication (NFC)

The NFC Forum is involved in the development and promotion of NFC. The 12 sponsor members of the NFC Forum include MasterCard International, Microsoft, Motorola, NEC, Nokia, Panasonic, Philips, Renesas, Samsung Electronics, Sony, Texas Instruments and Visa

- Capacity: 212 kbps over a distance from 0 to 20 centimeters over the 13.56 Mhz frequency range
- The NFC standard is based on RFID technology
- Applications suggested for NFC include ticketing, payment and gaming.
- Support for a passive mode of communication leads to savings on battery power

Annex Z
Bibliography

- A DNP3 Protocol Primer, DNP.org, http://www.dnp.org/AboutUs/DNP3%20Primer%20Rev%20A.pdf

. - DNP3 Overview, Triangle Microworks, http://www.trianglemicroworks.com/documents/DNP3_Overview.pdf

The world market for substation automation and integration programs in electric utilities: 2011-13, Volume 1, North American Market, Newton-Evans Research Company, http://www.dnp.org/Lists/Announcements/Attachments/6/Newton%20Evans%20NA%20SSA%202011V1.pdf

History

Approval history		
V.1.x.x	xx-mmm-2013	<Approved Version>

Draft history (to be removed on publication)		
V.0.0.1	dd Mmm 2013	Skeleton Draft
V.0.1.1	08 Aug 2013	Output Draft - PRO WG3 at TP#6 - Toronto - Including Contributions: oneM2M-PRO-2013-023, -024R01, -029R02
V0.1.2	08 Aug 2013	Revised Output Draft - PRO WG3 at TP#6 - Toronto: adding input from oneM2M-PRO-2013-025R03,
V0.2.0	02 Sep 2013	PRO WG3 Agreed output from TP#6
V0.2.1	25 Sep 2013	Output from PRO WG3 meeting 11 Sep 2013, including oneM2M-PRP-2013-0034R02, oneM2M-PRP-2013-0041R01
V0.2.2	30 Sep 2013	Output from PRO WG3 meeting 25 Sep 2013, including V0.2.1 and oneM2M-PRO-2013-0044R01, oneM2M-PRO-2013-0045, and oneM2M-PRO-2013-0046R01.
V0.2.3	23 Oct 2013	Output Draft - PRO WG3 at TP#7 - Sophia Antipolis; Including Contributions: oneM2M-PRO-2013-0050R02, 0051R01 (w/text from 0058), -0053R02, -0054R03, -0056R02, and -0062. Added references and acronyms.
V0.3.0	03 Nov 2013	PRO WG3 Agreed output from TP#7
V0.3.1	05 Nov 2013	Output draft from PRO 7.1 WG3 meeting 30 Oct 2013, including oneM2M-PRO-2013-0066R01
V0.3.2	12 Nov 2013	Output draft from PRO 7.2 WG3 meeting 06 Nov 2013, including oneM2M-PRO-2013-0071R01. Additional references and acronyms.
V0.3.3	27 Nov 2013	Output draft from PRO 7.4, 20 Nov and PRO 7.5, 27 Nov 2013, including oneM2M-PRO-2013-0082R01 and oneM2M-PRO-2013-0086
V0.3.4	12 Dec 2013	Output Draft - PRO WG3 at TP#8 - Miyazaki; Including Contributions: oneM2M-PRO-2013-0084R01, -0087, and -0091R02
V0.4.0	06 Jan 2014	PRO WG3 Agreed output from TP#8
V0.4.1	19 Feb 2014	Output Draft - PRO WG3 at TP9 – Mobile AL; Including Contributions: PRO-2014-0030-Bluetooth_revisions
V0.5.0	07 Mar 2014	PRO WG3 Agreed output from TP9
V0.5.1	10 Apr 2014	Output Draft - PRO WG3 at TP10 – Berlin, including PRO-2014-0121R01 from interim meetings PRO 9.2 and PRO 9.3: PRO-2014-0128R03 (ISA100), -0132R01 (WirelessHART), -0135 (DNP3), -0136 (Modbus), -0137 (XMPP), -0138 (CoAP), -0146 (Bluetooth), -0163 (HTTP) Applied template to 6.4.x, normalized NOTE in all 6.x.12
V0.5.2	22 April 2014	Output Draft - after mailing list review post PRO WG3 at TP10, with editorial changes suggested by Secretariat

V0.6.0	22 April 2014	PRO WG3 Agreed output from TP10
V0.6.1	20 May 2014	Output Draft from PRO 10.2; Including contribution PRO-2014-0192
V0.6.2	27 May 2014	Output Draft from PRO 10.7; Including contribution PRO-2014-0209R01 with meeting comments
V0.6.3	28 May 2014	Revised Output Draft to include contribution PRO-2014-0127R01, from PRO 9.4, which was agreed but not previously included.
V0.6.4	10 June 2014	Output Draft from PRO 11.0 to include contributions PRO-2014-0234R01 and PRO-2014-0248R01.
V0.7.0	12 June	Approved by WG3