International Telecommunication Union

ITU-T Y.4500.10

(03/2018)

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Internet of things and smart cities and communities – Frameworks, architectures and protocols

oneM2M – MQTT protocol binding

Recommendation ITU-T Y.4500.10

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Network control architectures and protocols	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	Y.3000–Y.3499
CLOUD COMPUTING	Y.3500–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	**Y.4400–Y.4549**
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.4500.10

oneM2M – MQTT protocol binding

Summary

Recommendation ITU-T Y.4500.10 specifies the binding of Mca and Mcc primitives (message flows), defined in the service layer core protocol, on to the MQTT transport protocol.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Y.4500.10	2018-03-01	20	11.1002/1000/13505

Keywords

oneM2M, MQTT, protocol, security, transport, primitive, message.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

NOTE – This Recommendation departs slightly from the usual editorial style of ITU-T Recommendations to preserve existing cross-referencing from external documents.

Table of Contents

Recommendation ITU-T Y.4500.10

oneM2M – MQTT protocol binding

1 Scope

This Recommendation specifies the binding of Mca and Mcc primitives (message flows) on to the message queuing telemetry transport (MQTT) protocol. It specifies how:

1) a common services entity (CSE) or application entity (AE) connects to MQTT;
2) an Originator (CSE or AE) formulates a request as an MQTT message, and transmits it to its intended receiver;
3) a receiver listens for incoming requests;
4) hat receiver can formulate and transmit a response.

The Recommendation contains oneM2M Release 2 specification – oneM2M MQTT protocol binding V2.4.1 and is equivalent to standards of oneM2M partners including ARIB, ATIS [b-ATIS.oneM2M.TS0010V241], CCSA [b-CCSA M2M-TS-0010], ETSI [b-ETSI TS 118 110], TIA, TSDSI [b-TSDSI STD T1.oneM2M TS-0010-2.4.1], TTA [b-TTAT.MM-TS.0010] and TTC [b-TTC TS-M2M-0010].

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Y.4500.1] Recommendation ITU-T Y.4500.1 (2018), *oneM2M – Functional architecture.*

[ITU-T Y.4500.4] Recommendation ITU-T Y. 4500.4 (2018), *oneM2M – Service layer core protocol specification*.

[ETSI TS 118 103] ETSI TS 118 103 V2.4.1 (2016), *oneM2M; Security solutions (oneM2M TS-0003 version 2.4.1 Release 2)*.

[IETF RFC 793] IETF RFC 793 (1981), *Transmission control protocol – DARPA internet program – Protocol specification*.

[IETF RFC 3986] IETF RFC 3986 (2005), *Uniform resource identifier (URI): Generic syntax*.

[IETF RFC 5246] IETF RFC 5246 (2008), *The transport layer security (TLS) protocol – Version 1.2*.

[IETF RFC 6455] IETF RFC 6455 (2011), *The WebSocket protocol*.

[OASIS MQTT] OASIS (2014), *MQTT* standard Version 3.1.1.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 application entity (AE): [b-ITU-T Y.4500.11] Represents an instantiation of application logic for end-to-end M2M solutions.

3.1.2 common services entity (CSE): [b-ITU-T Y.4500.11] Represents an instantiation of a set of common service functions of the M2M environments. Such service functions are exposed to other entities through reference points.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 originator: Actor that initiates a request.

NOTE – An originator can either be an application or a common services entity.

3.2.2 receiver: Actor that receives the request.

NOTE – A receiver can be a common services entity or an application.

3.2.3 resource: Entity in the oneM2M system that is uniquely addressable, e.g., by the use of a uniform resource identifier (URI).

NOTE – A resource can be accessed and manipulated by using the specified procedures.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ADN	Application Dedicated Node
ADN-AE	Application Dedicated Node-Application Entity
AE	Application Entity
ASN	Application Service Node
CBOR	Concise Binary Object Representation
CRUD	Create-Read-Update-Delete
CSE	Common Service Entity
HTTP	Hypertext Transfer Protocol
ID	Identifier
IN	Infrastructure Node
IN-AE	Infrastructure Node- Application Entity
IN-CSE	Infrastructure Node-Common Service Entity
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
M2M	Machine to Machine
MN	Middle Node
MN-CSE	Middle Node-Common Service Entity

MQTT	Message Queuing Telemetry Transport
NAT	Network Address Translation
QoS	Quality of Service
SP	Service Provider
TCP	Transmission Control Protocol
TLS	Transport Level Security
URI	Uniform Resource Identifier
XML	Extensible Markup Language

5 Conventions

The keywords "shall", "shall not", "may", "need not", "should", "should not" in this Recommendation are to be interpreted as follows.

Shall/shall not:

Requirements:

1) effect on this Recommendation: this Recommendation needs to describe the required feature (i.e., specify a technical solution for the requirement);
2) effect on products: every implementation (M2M solution that complies with this this Recommendation) must support it;
3) effect on deployments: every deployment (M2M service based on this Recommendation) must use the standardized feature where applicable – otherwise interoperability problems with other services could arise for example.

Should/should not:

Recommendation

1) effect on this Recommendation: this Recommendation needs to describe a solution that allows the presence and the absence of the feature;
2) effect on products: an implementation may or may not support it, however support is recommended;
3) effect on deployments: a deployment may or may not use it, however usage is recommended.

May/need not:

Permission/option

1) effect on this Recommendation: this Recommendation needs to describe a solution that allows the presence and the absence of the required feature;
2) effect on products: an implementation may or may not support it;
3) effect on deployments: A deployment may or may not use it.

6 Introduction

6.1 Use of MQTT

This binding makes use of MQTT to provide reliable two-way communications between two parties (AEs and CSEs). It uses the following features of MQTT.

– Durable sessions, providing store and forward in cases where network connectivity is not available.

– MQTT's "QoS 1" message reliability level. This provides reliability without incurring the overhead implied by QoS 2.
– Network address translation (NAT) traversal [neither of the two parties is required to have prior knowledge of the other party's Internet protocol (IP) address].
– Dynamic topic creation and wild-carded subscription filters.

It does not use the following features:

– one-to-many publish or subscribe;
– retained messages;
– will messages;
– QoS 0 or QoS 2 message reliability levels.

6.2 Binding overview

6.2.1 Introduction

The MQTT protocol binding specifies how the Mca or Mcc request and response messages are transported across the MQTT protocol. Both communicating parties (AEs and CSEs) typically make use of an MQTT client library, and the communications are mediated via the MQTT server. There is no need for the client libraries or the server to be provided by the same supplier, since the protocol they use to talk to each other is defined by the MQTT specification [OASIS MQTT].

Furthermore, the binding does not assume that the MQTT client libraries or server implementations are necessarily aware that they are being used to carry Mca, Mcc or any other oneM2M-defined primitives.

The binding is defined in terms of the MQTT protocol flows that take place between the client libraries and the MQTT server in order to effect the transport of an Mca or Mcc message.

There are two scenarios depending on the location of MQTT server: MQTT server co-located within a node and MQTT server located independently from nodes.

6.2.2 Scenarios

6.2.2.1 MQTT server co-located scenario

Figure 6.2.2.1-1 shows a protocol segment view of the MQTT server co-located scenario, in which all oneM2M nodes [application dedicated node (ADN), application service node (ASN), middle node (MN), infrastructure node (IN)] include one or more MQTT clients. MQTT servers are provided within MNs and INs.

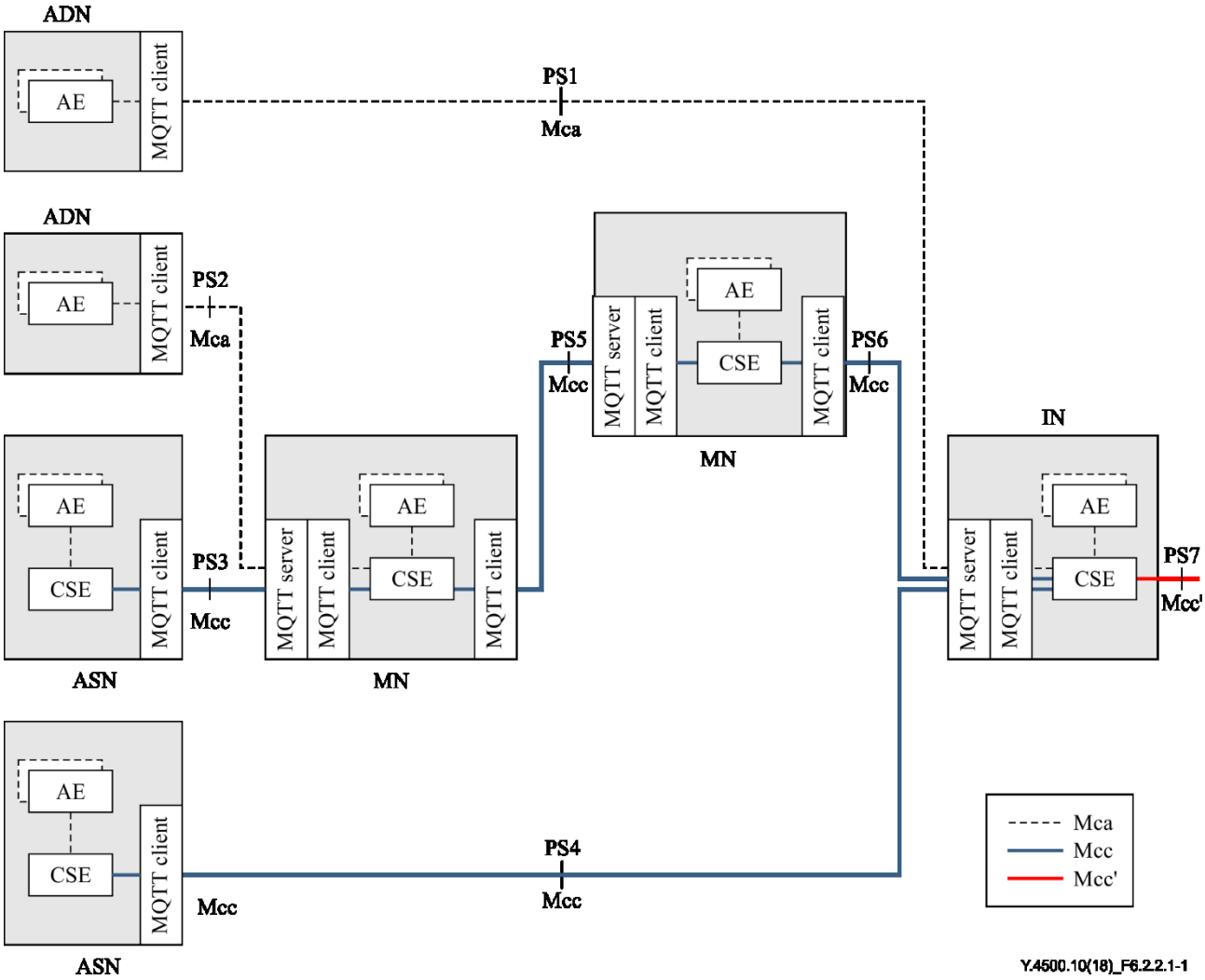

Figure 6.2.2.1-1 – MQTT server co-located scenario

In this scenario, the protocol segments are as listed in Table 6.2.2.1-1.

Table 6.2.2.1-1 – Protocol segment for MQTT server co-located scenario

Protocol Segment	oneM2M Message Transported	MQTT Interaction
PS1	Mca (AE of ADN to CSE of IN)	Client in ADN to Server in IN
PS2	Mca (AE of ADN to CSE of MN)	Client in ADN to Server in MN
PS3	Mcc (CSE of ASN to CSE of MN)	Client in ASN to Server in MN
PS4	Mcc (CSE of ASN to CSE of IN)	Client in ASN to Server in IN
PS5	Mcc (CSE of MN to CSE of MN)	Client in MN to Server in MN
PS6	Mcc (CSE of MN to CSE of IN)	Client in MN to Server in IN
PS7	Mcc' (CSE of IN to CSE of IN)	Client in IN to Server in IN

6.2.2.2 MQTT server independently located scenario

Figure 6.2.2.2-1 shows a protocol segment view in which the MQTT server is located independently from the oneM2M nodes. In this scenario, all oneM2M nodes (ADN, ASN, MN, IN) include one or more MQTT clients. MQTT servers exist independently, which means the servers are located outside the nodes.

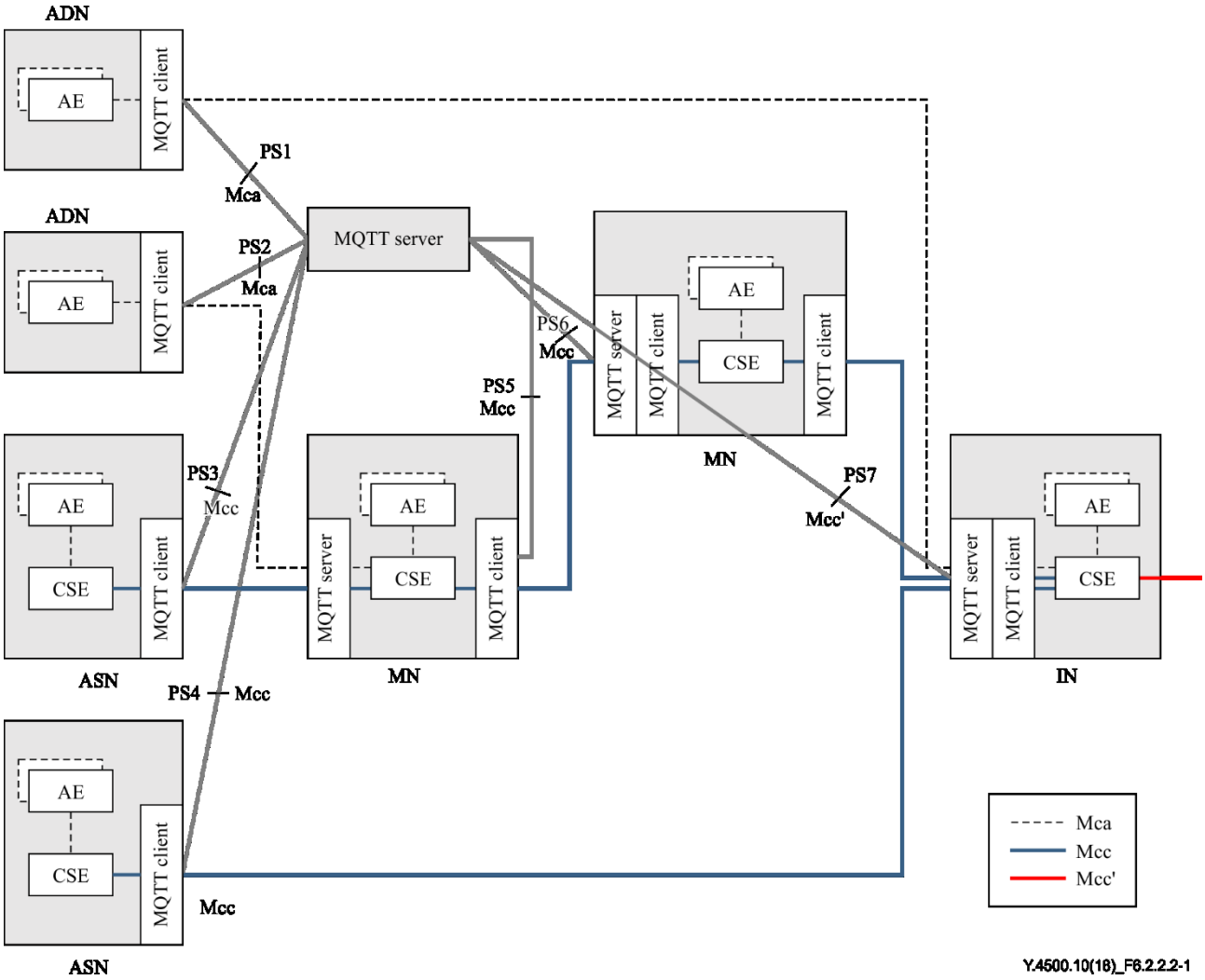

Figure 6.2.2.2-1 – MQTT server independently located scenario

In this scenario, the protocol segments are as listed in Table 6.2.2.2-1.

Table 6.2.2.2-1 – Protocol segment for MQTT server independently located scenario

Protocol Segment	oneM2M Message Transported	MQTT Interaction
PS1	Mca (AE of ADN to CSE of IN)	Client in ADN to Server
PS2	Mca (AE of ADN to CSE of MN)	Client in ADN to Server
PS3	Mcc (CSE of ASN to CSE of MN)	Client in ASN to Server
PS4	Mcc (CSE of ASN to CSE of IN)	Client in ASN to Server
PS5	Mcc (CSE of MN to CSE of MN) Mcc (CSE of MN to CSE of ASN) Mca (CSE of MN to AE of ADN)	Client in MN to Server
PS6	Mcc (CSE of MN to CSE of MN) Mcc (CSE of MN to CSE of IN)	Client in MN to Server
PS7	Mcc (CSE of IN to CSE of MN) Mcc (CSE of IN to CSE of ASN) Mca (CSE of IN to AE of ADN)	Client in IN to Server

Clauses 6.2.3.1 to 6.2.3.4 show the four configurations in which the MQTT binding can be used in the co-located scenario, followed by similar configurations in the independently located scenario.

NOTE – Other configurations are possible, but they currently lie outside the scope of this Recommendation.

6.2.3 Configurations

6.2.3.1 AE to IN

This configuration, illustrated in Figure 6.2.3.1-1, allows an AE to connect to an IN via MQTT.

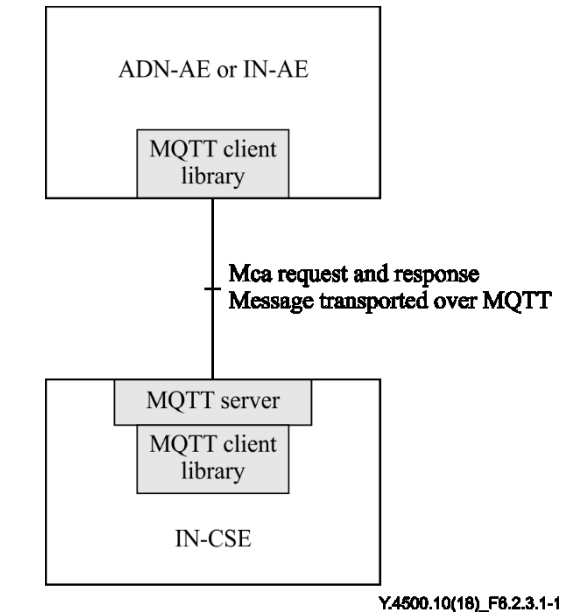

Figure 6.2.3.1-1 – Using MQTT between AE and IN-CSE

The MQTT server is co-located with the infrastructure node-common service entity IN-CSE (where the CSE resides in the IN) and allows connection of the application dedicated node-application entities (ADN-AEs; typically devices) or infrastructure node-application entities (IN-AEs; where the AE is registered with the CSE in the IN). The MQTT server can store and forward messages if there is a gap in the connectivity with the devices. Note that the AEs each establish their own separate transmission control protocol/Internet protocol (TCP/IP) connection with the MQTT server. Thus the server shall have an accessible IP address, but AEs need not have one.

6.2.3.2 AE to MN

This configuration, illustrated in Figure 6.2.3.2-1, allows an ADN-AE to connect to an IN via MQTT.

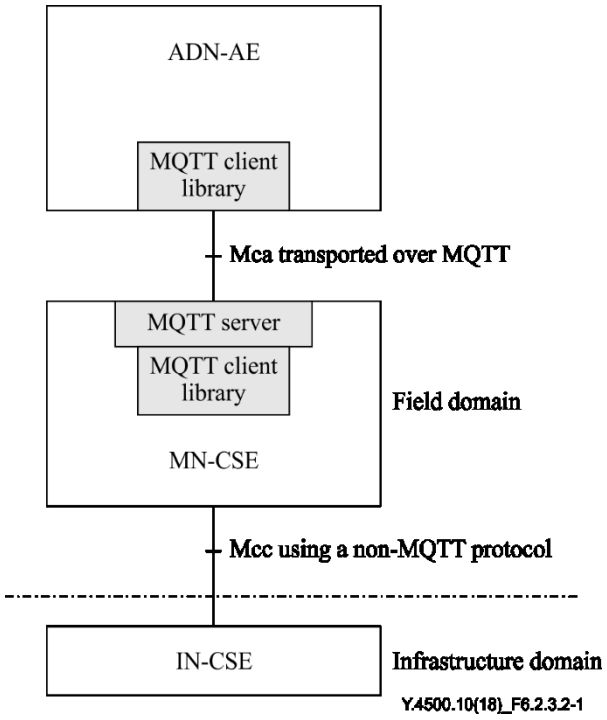

Figure 6.2.3.2-1 – Using MQTT between AE and MN-CSE

This configuration is very similar to the AE-IN configuration shown in clause 6.2.3.1, except that the MQTT server is hosted on the MN rather than the IN. Onwards connection to the IN-CSE is via a different transport protocol.

6.2.3.3 MN to IN

This configuration, illustrated in Figure 6.2.3.3-1, allows an MN to connect to an IN via MQTT.

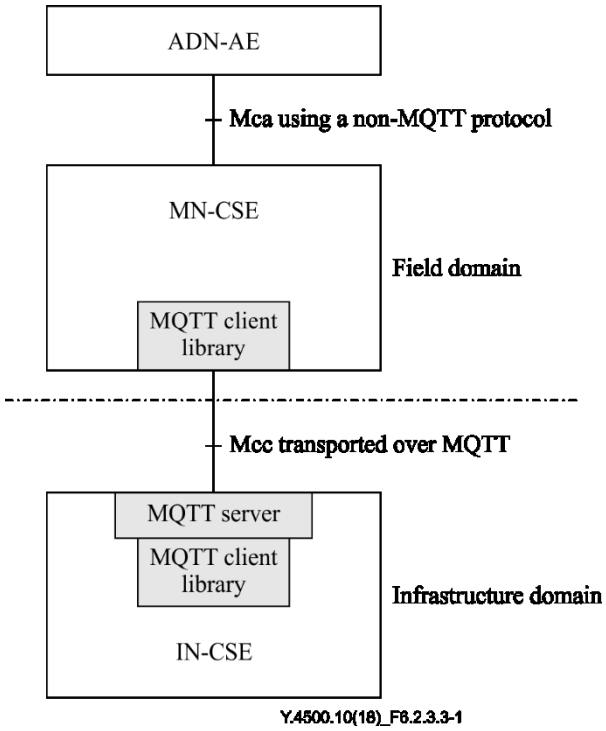

Figure 6.2.3.3-1 – Mcc using MQTT between MN and IN

The MQTT server is co-located with the IN-CSE and allows connection of the MNs (typically in-field gateway boxes). It can store and forward messages if there is a gap in the connectivity with the gateways. Note that the MNs each establish their own separate TCP/IP connections with the MQTT server. Thus the server shall have an accessible IP address, but MNs need not have one.

6.2.3.4 AE to MN to IN

This configuration, illustrated in Figure 6.2.3.4-1, is a combination of those specified in clauses 6.2.3.2 and 6.2.3.3.

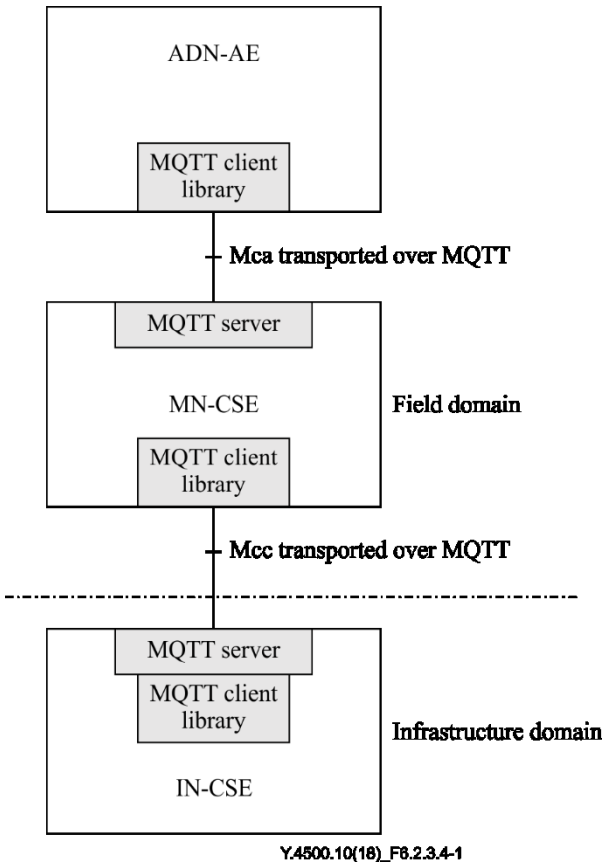

Figure 6.2.3.4-1 – Mca and Mcc both using MQTT

In this configuration, the two MQTT servers are independent from each other (i.e., they do not have a shared topic space). Any interactions between the ADN-AE and the IN-CSE are mediated by the middle node-common service entity (MN-CSE; where the CSE resides in the MN).

6.2.3.5 AE to IN (independent scenario)

This configuration, illustrated in Figure 6.2.3.5-1, allows an AE to connect to an IN via MQTT.

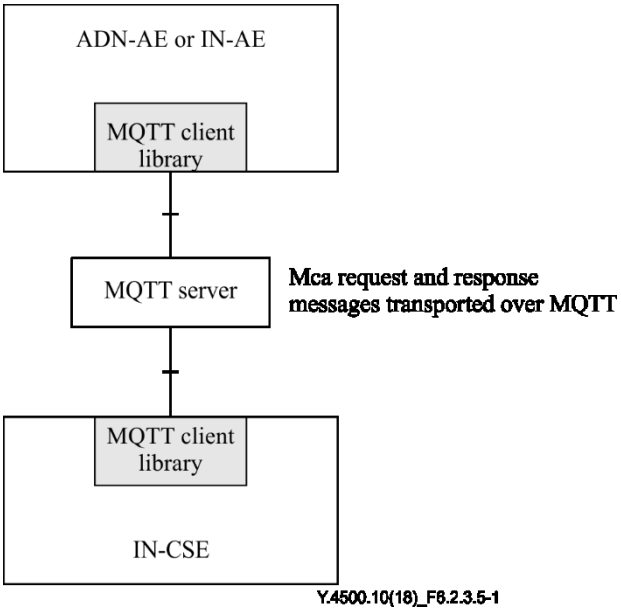

Figure 6.2.3.5-1 – Using MQTT between AE and IN-CSE

The MQTT server is an independent entity, located outside the nodes. In order to deliver Mca messages, MQTT clients within ADN-AE/IN-AE and IN-CSE connect to the MQTT server. After clients establish a TCP/IP connection with the MQTT server, Mca messages between ADN-AE/IN-AE and IN-CSE can be transported via the MQTT server.

6.2.3.6 AE to MN (independent scenario)

This configuration, illustrated in Figure 6.2.3.6-1, allows an ADN-AE to connect to an IN via MQTT.

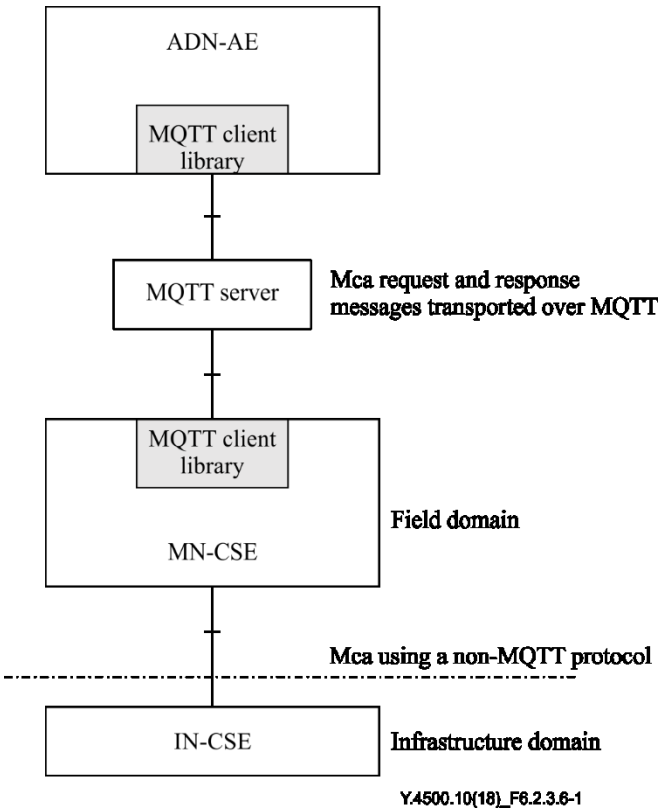

Figure 6.2.3.6-1 – Using MQTT between AE and MN-CSE

In this configuration, the MQTT server is an independent entity, located outside the nodes. MQTT clients within ADN-AE and MN-CSE are connected to the MQTT server, and the MQTT server stores and forwards the Mca messages between ADN-AE and MN-CSE. In addition, Figure 6.2.3.6-1 shows that the onwards connection to the IN-CSE is via a different transport protocol.

6.2.3.7 MN to IN (independent scenario)

This configuration, illustrated in Figure 6.2.3.7-1, allows an MN to connect to an IN via MQTT.

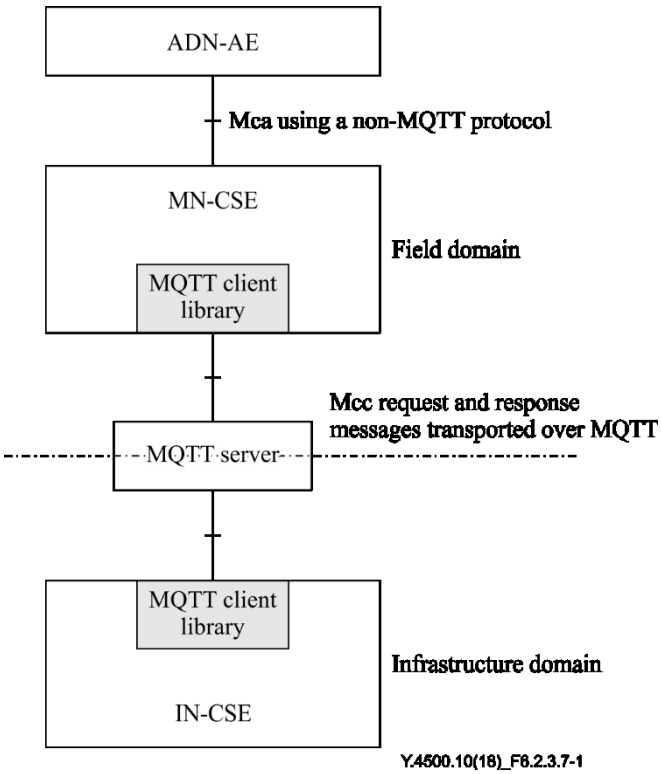

Figure 6.2.3.7-1 – Mcc using MQTT between MN and IN

In this configuration, the MQTT server is an independent entity, located outside nodes. Mcc message delivery between MN-CSE and IN-CSE are performed via the independently located MQTT server. As introduced in clauses 6.2.3.1 to 6.2.3.6, in order to send messages, each MQTT client within MN-CSE and IN-CSE connects to the MQTT server and Mcc messages are transported via the MQTT server.

6.2.3.8 AE to MN to IN (independent scenario)

This configuration, illustrated in Figure 6.2.3.8-1, is a combination of those specified in clauses 6.2.3.6 and 6.2.3.7.

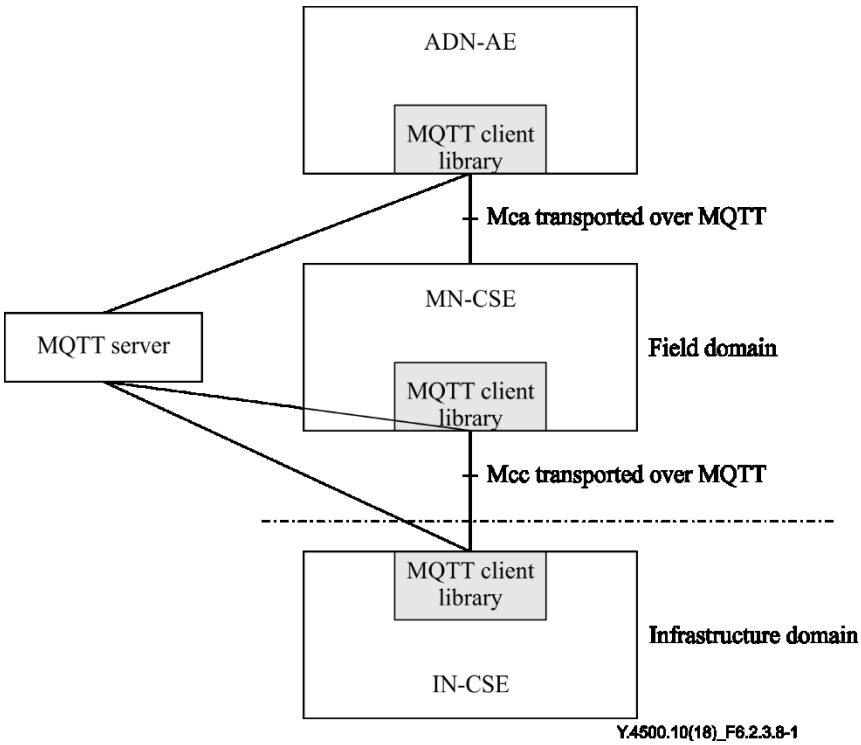

Figure 6.2.3.8-1 – Mca and Mcc both using MQTT

In this configuration, the MQTT clients of ADN-AE and MN-CSE and IN-CSE connect to the independently located MQTT server. Any interactions, such as Mca or Mcc message delivery among the ADN-AE and the MN-CSE and the IN-CSE, are mediated by the MQTT server.

7 Protocol binding

7.1 Introduction

In this clause, the use of MQTT is profiled and the key elements of the binding are defined, namely how:

1) a CSE or AE connects to MQTT;
2) an originator (CSE or AE) formulates a request as an MQTT message, and transmits it to its intended receiver;
3) a Receiver listens for incoming requests, and formulates and transmits a response.
4) the Mca and Mcc create-read-update-delete (CRUD) operations map to MQTT messages.

For more information on MQTT itself, see Appendix I or refer to the MQTT specification [OASIS MQTT].

7.2 Use of MQTT

MQTT includes reliability features that allow recovery from loss of network connectivity without requiring explicit involvement of the applications that are using it; however, to do this it requires an underlying network protocol that provides ordered, lossless, bi-directional connections. The MQTT specification [OASIS MQTT] does not mandate a particular underlying protocol, so this binding specification restricts the choice of underlying protocol: it shall be one of the following:

– raw TCP/IP [IETF RFC 793];
– TCP/IP with transport level security (TLS) [IETF RFC 5246];
– WebSocket [IETF RFC 6455] – either with or without the use of TLS.

7.3 Connecting to MQTT

In order to communicate, the two client parties (AE and CSE or CSE and CSE) shall connect to a common MQTT server. The MQTT server shall be hosted in one of the two nodes or shall exist as an independent external entity, following one of the two scenarios shown in clause 6.2.

Once each party has located the address of the MQTT server, it then connects to it using the standard MQTT CONNECT control packet.

An MQTT control packet consists of up to three parts: a fixed header, a variable header, and a payload as shown in Figure 7.3-1.

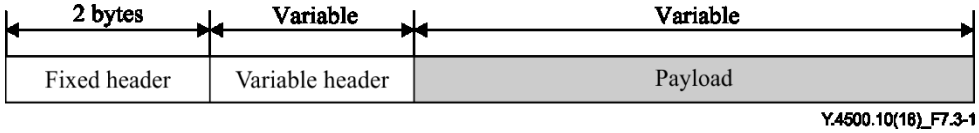

Figure 7.3-1 – Format of an MQTT control packet

7.3.1 Variable header of an MQTT CONNECT packet

A variable header for the MQTT CONNECT packet consists of four fields in the following order: `Protocol Name`, `Protocol Level`, `Connect Flags` and `Keep Alive` as shown in Figure 7.3.1-1.

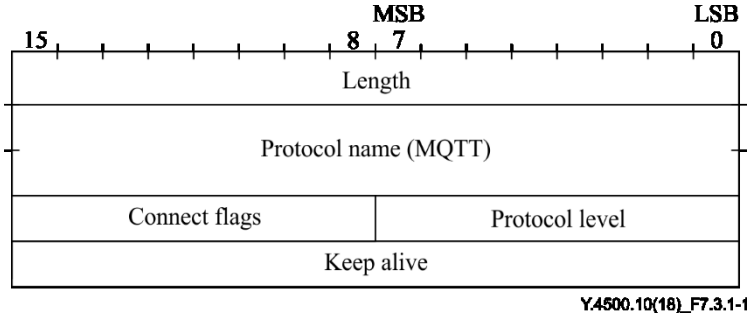

Figure 7.3.1-1 – Variable header of an MQTT CONNECT packet

The value of the `Protocol Name` field is "MQTT". The value of the `Protocol Level` field for the MQTT version 3.1.1 of the protocol is four. The `Connect Flags` is shown in Figure 7.3.1-2. The `Keep Alive` is a time interval measured in seconds.

Bit 7	Bit 6	Bit 5	Bit 3 ~ 4	Bit 2	Bit 1	Bit 0
User name flags	Password flag	Will retain	Will QoS	Will flag	Clean session	Reserved

Y.4500.10(18)_F7.3.1-2

Figure 7.3.1-2 – Connect Flags of variable header for an MQTT CONNECT packet

7.3.2 Payload of an MQTT CONNECT packet

A payload for the MQTT CONNECT packet is determined by the `Connect Flags` in the variable header. These fields may consist of `Client Identifier`, `Will Topic`, `Will Message`, `User Name`, `Password`.

Mandatory fields to establish MQTT session for oneM2M are:

– `Client Identifier`

Optional fields to establish MQTT session for oneM2M are:

– `User Name`

– `Password`

An example of a payload of an MQTT CONNECT Packet is shown in Figure 7.3.2-1.

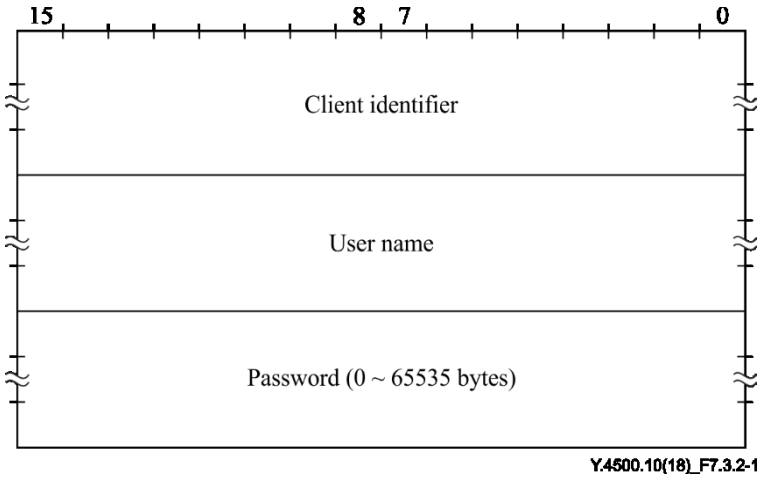

Figure 7.3.2-1 – Example of a payload of an MQTT CONNECT packet

7.3.3 Application of an MQTT CONNECT packet

The following additional considerations apply.

– Client Identifier: The CONNECT Packet contains a Client Identifier as described in clause 7.3.2. Client Identifiers have to be unique at least among all clients that connect to a given MQTT server instance (this is a requirement imposed by the MQTT protocol). This condition will be satisfied if an AE uses its AE-ID and a CSE uses its CSE-ID. See clause 7 of oneM2M TS-0001 [ITU-T Y.4500.1] for a discussion of these Identifiers (IDs). The prefix A: or C: shall be added to the ID to show whether it is an AE-ID or a CSE-ID as these ID spaces are not distinct. The AE-ID or CSE-ID may not be known during the initial registration process, in which case the client shall use some other appropriate unique ID.

– Connect Flags:
  – A client shall set the "Clean Session" flag in the MQTT CONNECT packet to false. This means that MQTT session state related to that client will be retained by the MQTT server in the event of a disconnection (deliberate or otherwise) of that client.
  – A client shall not set the "`Will Flag`", "`Will QoS`", or "`Will Retain`" so `Will Message` and `Will Topic` shall not be present in the payload.

– Keep Alive: A client may choose to provide a non-zero MQTT Keep Alive value or to provide a Keep Alive of 0 (this disables the MQTT Keep Alive).

– User Name and Password: The MQTT server may require that a client provide a user name and a password (or other credential). If the MQTT server authenticates by `user name` and `password`, the corresponding `user name` flag and `password` flag in the CONNECT shall be set to 1. For more information, see clause 8..

A client might choose to keep the MQTT connection open permanently (restarting it as soon as possible after any unforeseen connection loss), a client might choose to connect only when it wants to act as an originator, or it might choose to connect based on the <schedule> associated with a relevant oneM2M resource.

Once a client has connected to the MQTT server it can then communicate (subject to authorization policies) with any other client connected to its server. There is no need for it to create another connection if it wants to communicate with a different counter-party.

When a client determines that it no longer wishes to participate in an MQTT session with its MQTT server it shall perform the following steps:

– disconnect from that server, if it is currently connected;

– reconnect with the cleanSession flag set to true;

– disconnect again.

These steps delete any state that the MQTT server might be holding on behalf of the client.

7.4 Sending and receiving messages

7.4.1 Request and response messages

MQTT does not have a data model to describe or constrain the content of its application message payloads (to that extent it is similar to a TCP socket). Mca and Mcc request messages shall be serialized into extensible markup language (XML) or JavaScript object notation (JSON) or concise binary object representation (CBOR) following the serialization process defined in clause 7.5.

The packet type field in Figure 7.4.1.1-1 is used to define the MQTT control Packet type. It is a 4-bit field whose possible values are listed in Table 7.4.1-1. When a oneM2M request or response message is bound to MQTT, its packet type shall have value 3, i.e., the request or response message is delivered in an MQTT PUBLISH packet.

Table 7.4.1-1 – MQTT control packet types

Reserved	0	Reserved for future use
CONNECT	1	Client request to connect to server
CONNACK	2	Connect acknowledgement
PUBLISH	3	Publish message
PUBACK	4	Publish message acknowledgement
PUBREC	5	Publish received (QoS=2)
PUBREL	6	Publish release (QoS=2)
PUBCOMP	7	Publish complete (QoS=2)
SUBSCRIBE	8	Client subscribe request
SUBACK	9	Subscribe acknowledgement
UNSUBSCRIBE	10	Client unsubscribe request
UNSUBACK	11	Unsubscribe acknowledgement
PINGREQ	12	Ping request
PINGRESP	13	Ping response
DISCONNECT	14	Client disconnection request
Reserved	15	Reserved for future use

7.4.1.1 Fixed header of MQTT PUBLISH packet

The fixed header of the MQTT PUBLISH packet consists of `RETAIN`, `QoS Level`, `DUP flag`, `Packet Type`, `Remaining Length` fields as shown in Figure 7.4.1.1-1. The QoS `Level` represents the QoS level of the MQTT PUBLISH packet with possible values of 0, 1 or 2. However, since oneM2M messages are idempotent, the `QoS Level` should not be set to QoS 2.

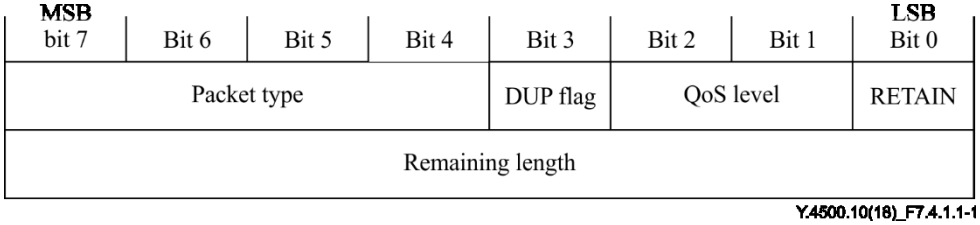

Figure 7.4.1.1-1 – Fixed header of an MQTT PUBLISH packet

NOTE – MQTT packets are subjected to a theoretical maximum message size of 256 MB, but it is good practice not to send packets that are bigger than 100 kB. If a larger amount of data needs to be sent, it should be segmented into multiple PUBLISH packets.

7.4.1.2 Variable header of an MQTT PUBLISH packet

The variable header for the MQTT PUBLISH packet consists of two fields in the following order: `Topic Name`, `Packet Identifier`.

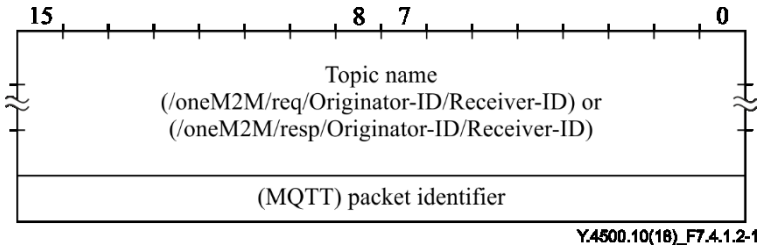

Figure 7.4.1.2-1 – Variable header for a PUBLISH packet

The `Topic Name` identifies the information channel to which payload data is published. The `Topic Name` for a oneM2M request message is specified in clause 7.4.2. The `Topic Filter` used to listen for and respond to a request is specified in clause 7.4.3.

The `Packet Identifier` field is only present in PUBLISH packets where the QoS level is 1 or 2.

7.4.1.3 Payload of an MQTT Control PUBLISH packet

The payload for an MQTT PUBLISH packet is a oneM2M request message or response message as specified in clause 7.5.

7.4.2 Topic name for requests

A request is transmitted by sending it as an MQTT PUBLISH packet to the MQTT server. The MQTT PUBLISH packet uses a `Topic Name` that identifies both the originator and the receiver of the request as follows.

– /oneM2M/req/<originator>/<receiver>/<type>
  – "oneM2M" is a literal string identifying the topic as being used by oneM2M
  – <originator> is the SP-relative-AE-ID (where SP is service provider) or SP-relative-CSE-ID of the entity that sends the request on this hop, omitting any leading "/"'s and replacing any other "/" characters with ":" characters
  – <receiver> is the SP-relative-AE-ID or SP-relative-CSE-ID of the receiver (AE, transit CSE or hosting CSE) on this hop, omitting any leading "/"'s and replacing any other "/" characters with ":" characters
  – "req" is a literal string identifying this as a request
  – <type> is "xml", "json" or "cbor" indicating the MQTT payload data type as described in clause 7.5.4

7.4.3 Listening for and responding to a request

A receiver listens for requests arriving via MQTT by subscribing using a wildcarded `Topic Filter` of the following form:

– /oneM2M/req/+/<receiver>
  – "oneM2M" is a literal string identifying the topic as being used by oneM2M
  – + is a wildcard that matches any entity

- <receiver> is the SP-relative-AE-ID or SP-relative-CSE-ID of the receiver (AE, transit CSE or hosting CSE), omitting any leading "/"'s and replacing any other "/" characters with ":" characters
- "req" is a literal string identifying this as a request.

When it receives a request, the receiver shall perform the core transport operations associated with the request, including any access control policy checks. In particular, the receiver shall check the request expiration timestamp (if any) contained in the request, since it is possible that time might have passed while the message was being stored by MQTT.

The receiver transmits a response by sending an MQTT PUBLISH packet to a response topic. This takes the following form.

– /oneM2M/resp/<originator>/<receiver>
  - "oneM2M" is a literal string identifying the topic as being used by oneM2M
  - <receiver> is the SP-relative-AE-ID or SP-relative-CSE-ID of the receiver (AE, transit CSE or hosting CSE), omitting any leading "/"'s and replacing any other "/" characters with ":" characters
  - <originator> is the SP-relative-AE-ID or SP-relative-CSE-ID of the entity that sent the corresponding request, omitting any leading "/"'s and replacing any other "/" characters with ":" characters
  - "resp" is a literal string identifying this as a response

The originator shall subscribe to this `Topic` (either explicitly or using a wildcarded filter) in order to see the response.

The payload of the MQTT PUBLISH packet is used to carry the response primitive, as described in clause 7.5.

7.4.4 Initial registration

In some security scenarios, an originator might not initially know its AE-ID or CSE-ID. Initial registration exchanges can use the communication pattern described in clauses 7.4.1 and 7.4.2 except that they use topics containing a credential ID rather than an AE-ID or CSE-ID, as follows.

– /oneM2M/reg_req/<originator>/<receiver>
  - "oneM2M" is a literal string identifying the topic as being used by oneM2M
  - <originator> is the Credential-ID. Any "/" characters embedded in the ID shall be replaced with ":" characters
  - <receiver> is the SP-relative-CSE-ID of the receiver (transit or hosting CSE) specified in the corresponding request, omitting any leading "/"
  - "reg_req" is a literal string identifying it as a registration request

and

– /oneM2M/reg_resp/<originator>/<receiver>
  - "oneM2M" is a literal string identifying the topic as being used by oneM2M
  - <originator> is the Credential-ID of the originator in the corresponding request. Any "/" characters embedded in the ID shall be replaced with ":" characters
  - <receiver> is the SP-relative-CSE-ID of the receiver (transit or hosting CSE) in the corresponding request, omitting any leading "/"
  - "reg_resp" is a literal string identifying it as a registration response.

7.4.5 Request/response message flow

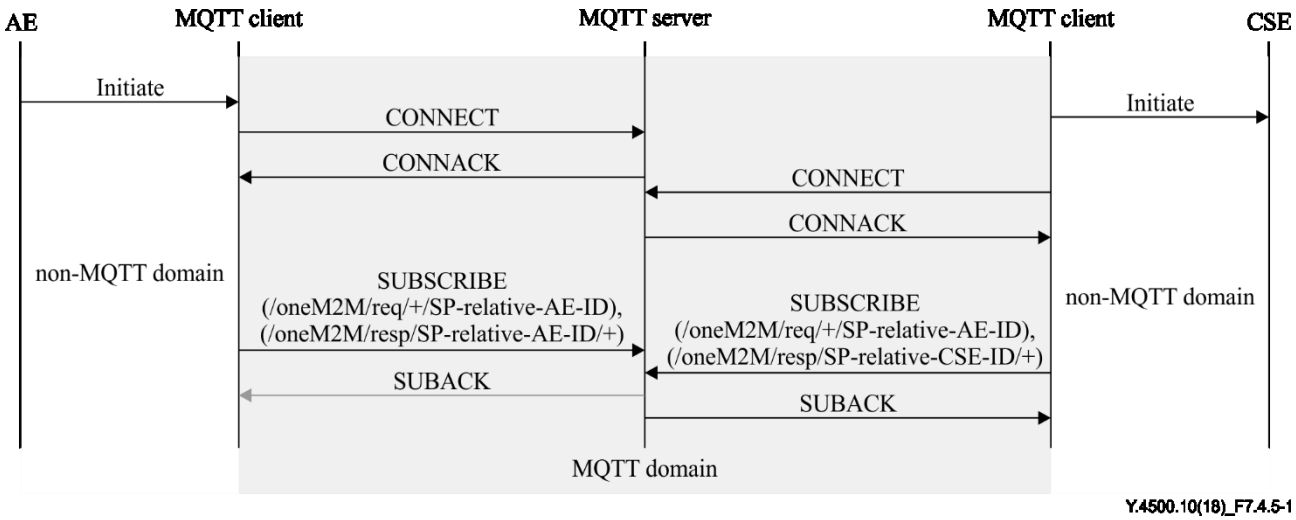

Figure 7.4.5-1 – Initiating process in MQTT binding

In the MQTT protocol, each client shall subscribe to the MQTT server to receive messages. As shown in Figure 7.4.5-1, the AE or CSE initiates the MQTT binding process by trying to connect to the MQTT server, as described in clause 7.3.

After each MQTT client successfully connects to the server, it shall subscribe to the MQTT server. The `Topic Filter`s to which each MQTT client subscribes are "/oneM2M/req/+/<receiver>" (to receive requests) and "/oneM2M/resp/<originator>/+" (to receive replies) where <receiver> and <originator> are both set equal to the ID (SP-relative-AE-ID or SP-relative-CSE-ID as appropriate).

Accordingly the plus sign ('+' U+002B) wildcard and the SP-relative-AE-ID or SP-relative-CSE-ID are used in the `Topic Filter` within the MQTT SUBSCRIBE Packet. This enables the MQTT client to receive the PUBLISH packets whose target it is. Therefore, through this process, the AE or CSE receives messages if the request/response messages are published to "/oneM2M/req/<originator>/<receiver>" or "/oneM2M/resp/<originator>/<receiver>".

As an example, Figure 7.4.5-2 illustrates request or response message delivery over the MQTT protocol between originator and receiver via the Mca/MCC reference point in oneM2M. The message flow is as follows.

In this flow, the originator wants to send a request message to the receiver. The originator's MQTT client library sends an MQTT PUBLISH packet to the MQTT server with "/oneM2M/req/ SP-relative-AE-ID/SP-relative-CSE-ID" as the `Topic Name`.

The MQTT PUBLISH packet shall include {"op", "fr", "to", "rqi"} and any optional parameters in accordance with the operation (CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY) as specified in clause 7.2.1.1 of oneM2M TS-0004 [ITU-T Y.4500.4] in its payload.

When the MQTT server receives the MQTT PUBLISH packet from the MQTT client, the server refers to the `Topic Name` and delivers the message to the intended MQTT client. Finally, the MQTT client library delivers the message to the receiver.

After that, the Receiver builds a `Topic Name` for the response message of the form "/oneM2M/resp/Originator-ID/Receiver-ID". The payload of the MQTT PUBLISH packet shall include {"rsc", "rqi"} and any optional message parameters in accordance with operation (CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY) as specified in clause 7.2.1.2 of oneM2M TS-0004 [ITU-T Y.4500.4] in its payload.

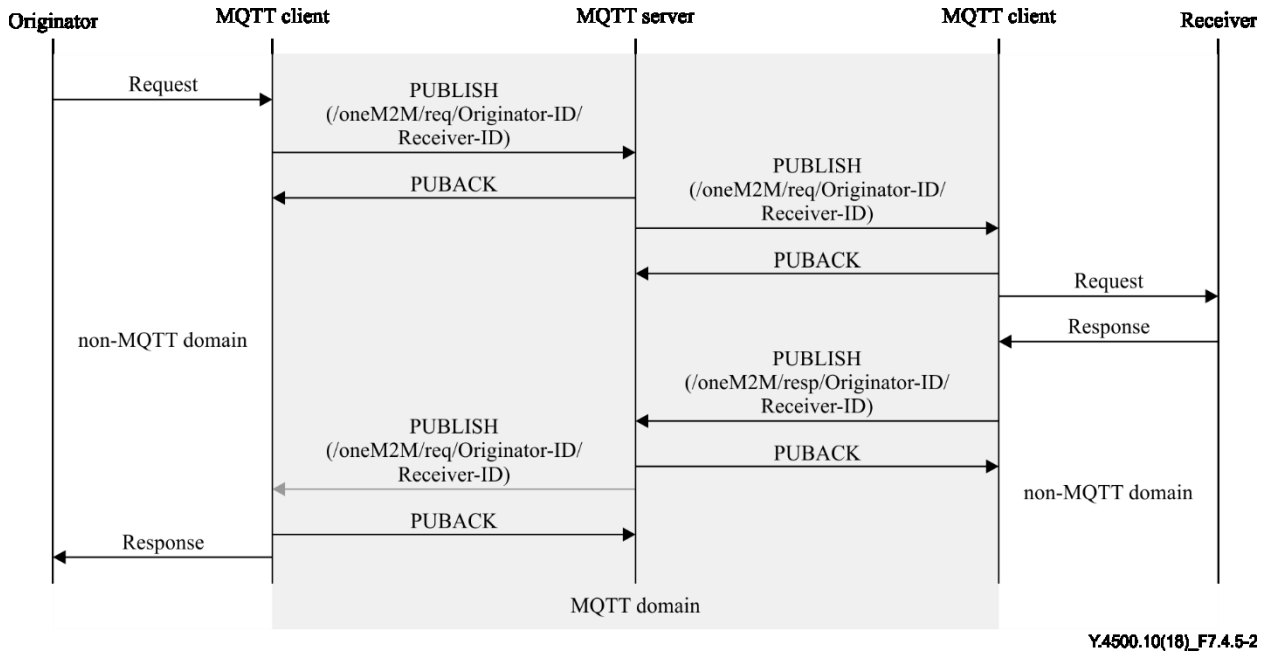

Figure 7.4.5-2 – Request or response message delivery over MQTT

7.5 Primitive mapping

7.5.1 Request primitives

A oneM2M request primitive is made up of a number of control parameters and (optionally) a content part. All the parameters in these parts are serialized into the payload of an MQTT PUBLISH packet, using the rules given in clause 8 of oneM2M TS-0004 [ITU-T Y.4500.4] applied to the m2m:requestPrimitive defined in clause 6.4.1 of oneM2M TS-0004 [ITU-T Y.4500.4]. See Figure 7.5.1-1.

All the parameters that are present in the primitive shall be serialized, in particular the request shall contain the mandatory parameters such as ***Operation, To, From, Request Identifier*** as specified in clause 8.1.2 of oneM2M TS-0001 [ITU-T Y.4500.1] and clause 7.1.1.1 of oneM2M TS-0004 [ITU-T Y.4500.4].

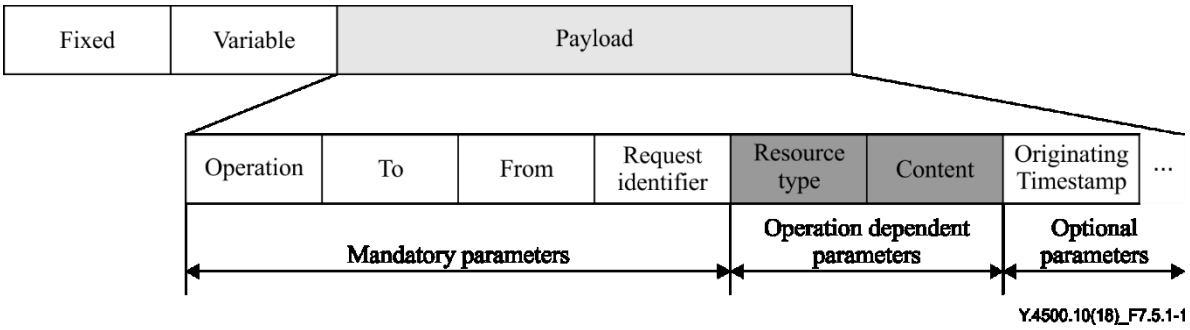

Figure 7.5.1-1 – MQTT request example

An example of an MQTT request message serialized using JSON is:

```
{"op": 1, "to": "//xxxxx/2345", "fr": "//xxxxx/99", "rqi": "A1234", "ty": 18, "pc":
{"m2m:sch":{"rn":"schedule1", "se":{"sce":["* 0-5 2,6,10 * * * *"]}}}, "ot": 20150910T062032}
```

– op: short name of ***Operation*** parameter specified as m2m:operation in oneM2M TS-0004 [ITU-T Y.4500.4]

– to: short name of ***To*** parameter specified either xs:anyURI [ITU-T Y.4500.4] or m2m:nhURI [ITU-T Y.4500.4]. It is a uniform resource identifier (URI) of the target resource.
– fr: short name of ***From*** parameter which is an ID of the originator e.g., either the AE or CSE.
– rqi: short name of ***Request Identifier*** specified as m2m:requestID [ITU-T Y.4500.4].
– ty: short name of ***Resource Type*** parameter specified as m2m:resourceType [ITU-T Y.4500.4].
– pc: short name of ***Content*** parameter specified in oneM2M TS-0004 [ITU-T Y.4500.4].
– ot: short name of ***Originating Timestamp*** parameter specified as m2m:timestamp [ITU-T Y.4500.4].

7.5.2 Response primitives

A oneM2M response primitive is serialized using the rules given in clause 8 of [ITU-T Y.4500.4] applied to m2m:responsePrimitive defined in clause 6.4.2 of oneM2M TS-0004 [ITU-T Y.4500.4].

In particular, each response primitive shall include the ***Response Status Code*** parameter to indicate success or failure of the operation and the ***Request Identifier*** parameter. See Figure 7.5.2-1.

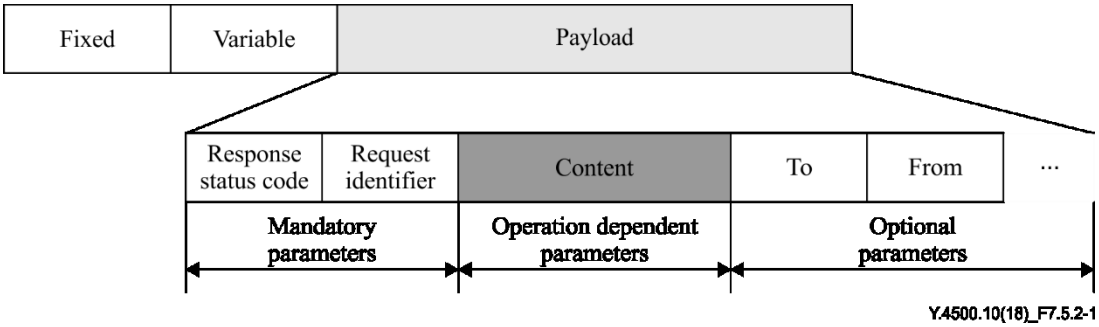

Figure 7.5.2-1 – MQTT response example

An example of an MQTT Response message serialized using JSON is:

```
{"rsc": 2000, "rqi": "A1234", "pc": {"m2m:sch":{"se":{"sce":["* 0-5 2,6,10 * * * *"]}}}, "to": "//xxxxx/2345",
"fr": "//xxxxx/99"}
```

– rsc: short name of ***Response Status Code*** parameter specified as m2m:responseStatusCode in oneM2M TS-0004 [ITU-T Y.4500.4].
– rqi: short name of ***Request Identifier*** specified as m2m:requestID in oneM2M TS-0004 [ITU-T Y.4500.4].
– pc: short name of ***Content*** parameter specified in oneM2M TS-0004 [ITU-T Y.4500.4].
– to: short name of ***To*** parameter specified either xs:anyURI [ITU-T Y.4500.4] or m2m:nhURI [ITU-T Y.4500.4]. It is an URI of the target resource.
– fr: short name of ***From*** parameter which is an ID of the Originator e.g., either the AE or CSE.

7.5.3 Serialization format negotiation

When sending a response primitive over MQTT, the receiver should use the same serialization that was used in the corresponding request primitive.

7.5.4 Content-type

An MQTT message payload contains a oneM2M request or response primitive that is serialized using the XML, the JSON or the CBOR encoding given in clause 8 of oneM2M TS-0004 [ITU-T Y.4500.4].

When an MQTT client publishes a message, it shall use a topic name to indicate the format of the payload included in a PUBLISH message. As defined in clause 8 of oneM2M TS-0004 [ITU-T Y.4500.4], the payload of request or response message shall include oneM2M primitives serialized using XML, JSON or CBOR.

The topic name takes this form:

– /oneM2M/req/<originator>/<receiver>/xml or /oneM2M/req/<originator>/<receiver>/json or /oneM2M/req/<originator>/<receiver>/cbor

– /oneM2M/resp/<originator>/<receiver>/xml or /oneM2M/resp/<originator>/<receiver>/json or /oneM2M/resp/<originator>/<receiver>/cbor

In order to receive a PUBLISH message with the topic name, each MQTT client shall subscribe to the topic name as follows:

– /oneM2M/req/+/<receiver>/#

– /oneM2M/resp/<originator>/#

7.6 Format used in *pointOfAccess* strings

oneM2M defines an MQTT URI format to be used in the *pointOfAccess* attributes in several entity resource types (e.g., <CSEBase>, <remoteCSE>, <AE>). A *pointOfAccess* attribute contains a list of one or more strings, each of which indicates a way in which that entity can be addressed. An entity can indicate support for MQTT by including strings in either or both of the following forms:

– mqtt://<authority>

– mqtts://<authority>

The <authority> component is defined in clause 3.2 of [IETF RFC 3986] and includes the host and optionally the port of the MQTT server that is to be used to access the entity in question.

The form with scheme mqtts: shall be used to show that the server requires the use of TLS when this particular point of access is being used.

If the <authority> does not contain a port component, then it shall be assumed that the MQTT ports are registered with the Internet Assigned Numbers Authority (IANA). The numbers are 1883 in the case of mqtt: and 8883 in the case of mqtts.

8 Security

8.1 Introduction

The MQTT servers authenticate the clients (both CSEs and AEs) that connect to them and authorize access to topics used for communication. The clients do not authenticate each other, instead they use the MQTT server to do this.

Background. The MQTT binding makes use of one or more MQTT servers to transport messages between AEs and CSEs (or between CSEs) as described in clause 6. The AE/CSEs both act as MQTT clients of an MQTT server that mediates delivery of messages between the two. As described in clause 7, the topic in the MQTT PUBLISH packet either indicates the identities of the originator and receiver or includes the credential of the originator and the identity of the receiver (if the originator has not yet been assigned an identifier).

Trust and the MQTT server. When the oneM2M binding to MQTT is used, some security functions are performed by the MQTT Server, as described in clauses 8.2 to 8.5. In particular, note the following.

1) The MQTT server authenticates the AEs and CSEs as they connect as MQTT clients. These MQTT clients themselves never directly authenticate the CSE or AE that is using another MQTT client – instead they trust the MQTT server to authenticate MQTT clients, and trust

the MQTT server to route the messages between the originator and receiver indicated in the topic.

2) The MQTT server enforces access control policies to ensure that unidentified or unauthorized clients are not able to publish messages to oneM2M topics or subscribe to receive messages from them.

8.2 Authorization

There are two levels of authorization in the oneM2M binding to MQTT.

– *The MQTT server is responsible for verifying identifiers, for routing messages to the expected CSE or AE, and providing the correct Credential-ID during initial registration*

The MQTT server is responsible for verifying that the Client Identifier field in a MQTT CONNECT packet matches the expected AE-ID, CSE-ID or Credential-ID.

The MQTT server is responsible for controlling those topics to which an MQTT client may subscribe and receive published MQTT packets, and those topics to which an MQTT client may publish MQTT packets.

Since the topic includes the CSE-ID of the receiver, the originator can trust that the MQTT packets are routed to and from the expected receiver. If the topic includes the CSE-ID or AE-ID of the originator, then the Receiver can trust that the MQTT packets are routed to and from the expected originator.

If the topic includes the Credential-ID of the originator (which should only occur at initial registration), then the receiver can use this Credential-ID to determine the CSE-ID or AE-ID or list of allowed AE-ID(s) that are to be used in assigning a CSE-ID or AE-ID to the Originator (as described in TS-0003 [ETSI TS 118 103]). This Credential-ID can be trusted to have been verified by the MQTT server.

– *The receiver is responsible for authorizing requests to specific resources, and assigning CSE-ID or AE-ID during initial registration*.

When the MQTT topic includes the CSE-ID or AE-ID of the originator, then the receiver is responsible for making access control decisions on requests to perform operations on specific resources hosted on the receiver. The access control decisions are dictated by the applicable accessControlPolicy resources and the CSE-ID or AE-ID of the originator (and other factors not relevant to the present discussion). This authorization process is as defined in the architecture specification [ITU-T Y.4500.1], the core protocol specification [ITU-T Y.4500.4] and security solutions specification [ETSI TS 118 103].

When the MQTT topic includes the Credential-ID of the originator (which should only occur at initial registration), then the receiver is responsible for assigning a CSE-ID or AE-ID to the originator (which may be dependent on the Credential-ID of the originator).

8.3 Authentication

An MQTT client and MQTT server shall apply TLS using any of the security association establishment frameworks in TS-0003 [ETSI TS 118 103].

Security association establishment frameworks provide mutual authentication of the MQTT client and MQTT Server. Security association establishment frameworks are described using two main entities: A and B. In the case of the oneM2M binding to MQTT, entity A is a CSE or AE using an MQTT client and entity B is an MQTT server.

NOTE – In TS-0003 [ETSI TS 118 103], entity A is described as establishing the CSE-ID of entity B as a result of security association establishment. The application to MQTT differs, because entity A establishes the identity of the MQTT server instead. However, the procedures are still applicable.

The remote security provision frameworks in TS-0003 [ETSI TS 118 103] may be applied to provision a symmetric key shared by a CSE/AE using an MQTT client and an MQTT server, with the MQTT server assuming the role of the enrolment target.

Identification of originator and receiver. TS-0003 [ETSI TS 118 103] describes a variety of approaches by which successful security association establishment results in entity B determining the CSE-ID or AE-ID or list of allowed AE-ID(s) for the CSE/AE using entity A. These approaches can also be used in the oneM2M binding to MQTT.

It is assumed that the MQTT server is configured with the information necessary to determine the CSE-ID of the receiver following successful security association establishment with the MQTT client of the receiver.

In some scenarios, the MQTT server can be configured with appropriate information to verify the CSE-ID or AE-ID of the originator. However, in cases where the originator has not yet been assigned its CSE-ID or AE-ID, and the MQTT server has also not been provided with the CSE-ID or AE-ID of the originator, then the receiver is responsible for determining the applicable CSE-ID or AE-ID. In these cases, the MQTT server forms a Credential-ID, identifying the credential used to authenticate the originator, and includes this in the topic when forwarding the initial registration request to the MQTT client of the receiver. The receiver extracts the Credential-ID, and the procedures in TS-0004 [ITU-T Y.4500.4] and TS-0003 [ETSI TS 118 103] determine the CSE-ID or AE-ID of the originator.

Password Field Authentication. This Recommendation does not specify use of the password field of the MQTT CONNECT control packet. Authentication is performed using the TLS mechanisms described in TS-0003 [ETSI TS 118 103].

8.4 Authorization by the MQTT server

This procedure describes how an MQTT server authorizes topics to which an MQTT client may subscribe. The machine to machine (M2M) SP is responsible for configuring the MQTT server with the relevant information used in these authorization decisions.

1) If the MQTT server determines that the MQTT client represents a CSE, and if the CSE has been assigned an SP-Relative-CSE-ID (which is denoted by <my-SP-Relative-CSE-ID>), then:
   a) the MQTT server authorizes the MQTT client to subscribe to the topic /oneM2M/req/+/<my-SP-Relative-CSE-ID>; and
   b) for the set of <Registree ID Stem> values corresponding to SP-Relative CSE-ID or AE-ID stem values of zero or more CSE(s) or AE(s) currently registered to this CSE (and known to the MQTT Server), the MQTT server authorizes the MQTT client to subscribe to the topics /oneM2M/resp/<Registree ID Stem>/<my-SP-Relative-CSE-ID>; and
   c) if this CSE is registered to a CSE and the SP-Relative-CSE-ID of this CSE is <Registrar-SP-Relative-CSE-ID > then the MQTT Server authorizes the MQTT Client to perform the following:
      i) To subscribe to /oneM2M/resp/<my-SP-Relative-CSE-ID>/<Registrar-SP-Relative-CSE-ID>, and
      ii) To publish to /oneM2M/resp/<my-SP-Relative-CSE-ID>/<Registrar-SP-Relative-CSE-ID>.
2) If the MQTT server determines that the MQTT client represents an AE that has been assigned an S-Type AE-ID stem equal to < AE-ID-Stem>, then:
   a) if the MQTT server determines that the AE is currently registered, and the registrar CSE of the AE has SP-Relative-CSE-ID equal to <Registrar-SP-Relative-CSE-ID> then the MQTT Server authorizes the MQTT Client to perform the following:
      i) to subscribe to

/oneM2M/resp/<AE-ID-Stem>/<Registrar-SP-Relative-CSE-ID>,

ii) to publish to

/oneM2M/req/<AE-ID-Stem>/<Registrar-SP-Relative-CSE-ID>,

iii) to subscribe to /oneM2M/req/<Registrar-SP-Relative-CSE-ID>/<AE-ID-Stem>, and

iv) to publish to /oneM2M/resp/<Registrar-SP-Relative-CSE-ID>/<AE-ID-Stem>;

b) otherwise, the MQTT server authorizes the MQTT client to perform the following:

i) to subscribe to /oneM2M/reg_resp/<AE-ID-Stem-Credential-ID>/+, and

ii) to publish to /oneM2M/reg_req/<AE-ID-Stem-Credential-ID>/+

where <AE-ID-Stem-Credential-ID> is generated from <AE-ID-Stem> as per TS-0003 [ETSI TS 118 103].

3) If the MQTT server determines that the MQTT client represents a AE with a C-Type AE-ID-Stem equal to <AE-ID-Stem> (which implies that the AE is registered), and the registrar CSE of the AE has an SP-Relative-CSE-ID equal to <Registrar-SP-Relative-CSE-ID>, then the MQTT Server authorizes the MQTT Client to perform the following:

a) to subscribe to /oneM2M/resp/<AE-ID-Stem>/<Registrar-SP-Relative-CSE-ID>;

b) to publish to /oneM2M/req/<AE-ID-Stem>/<Registrar-SP-Relative-CSE-ID>;

c) to subscribe to /oneM2M/req/<Registrar-SP-Relative-CSE-ID>/<AE-ID-Stem>; and

d) to publish to /oneM2M/resp/<Registrar-SP-Relative-CSE-ID>/<AE-ID-Stem>.

4) In all other cases, the MQTT server authorizes the MQTT client to perform the following:

a) to subscribe to /oneM2M/reg_resp/<Credential-ID>/+; and

b) to publish to /oneM2M/reg_req/<Credential-ID>/+

where <Credential-ID> is obtained from the security association establishment procedure as described in TS-0003 [ETSI TS 118 103].

8.5 General considerations

Implementors should take note of the security considerations listed in Chapter 5 of [OASIS MQTT].

Annex A

oneM2M specification update and maintenance control procedure

(This Annex forms an integral part of this Recommendation.)

The provisions of Annex L of [ITU-T Y.4500.1] regarding oneM2M specification update and maintenance control procedure shall apply to this Recommendation.

Appendix I

Overview of MQTT

(This appendix does not form an integral part of this Recommendation.)

I.0 Introduction

This annex provides some background information on MQTT that might be useful to a reader of the normative clauses of this Recommendation. See [OASIS MQTT] for the definitive source of information about the protocol itself.

I.1 MQTT features

MQTT is a lightweight publish or subscribe messaging transport protocol, particularly well suited to event-oriented interactions. It was specifically designed for constrained environments, such as those found in M2M and Internet of things (IoT) contexts where a small code footprint is required or network bandwidth is at a premium.

MQTT includes reliability features that allow recovery from loss of network connectivity without requiring explicit involvement of the applications that are using it; however, it does require an underlying network protocol that provides ordered, lossless, bi-directional connections.

The features of MQTT include the following.

- The use of the publish or subscribe message pattern that provides one-to-many message distribution and decoupling of applications. This is described further in clause I.3.1.
- Bidirectional communications. An entity can subscribe to receive messages without having a reliable IP address. This could be used to allow unsolicited requests to be sent to a receiver, or an asynchronous response to be sent to an originator, where the originator or receiver does not have an externally accessible IP address. It thus eliminates the need for long polling and can reduce the need for triggering.
- A messaging transport that is agnostic to the content of the payload. The message payload can be text or binary.
- A session concept that can survive loss of network connectivity and can persist across multiple consecutive network connections. Messages can be stored and subsequently forwarded when connectivity is restored.
- Three levels of reliability (referred to as "qualities of service") for message delivery within a Session as follows.
  - "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
  - "At least once", where message arrival is ensured, but duplicates might occur. This is best suited to messages that have idempotent semantics.
  - "Exactly once", where message arrival is ensured exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchange designed to minimize network traffic, with consequent additional savings on battery power when compared to the hypertext transfer protocol (HTTP).
- A retained message option, allowing new subscribers to get the last message to have been published on a topic prior to their subscription.

– A mechanism to notify interested parties when an abnormal disconnection occurs.

I.2 MQTT implementations

Like HTTP, the MQTT protocol is asymmetric in that it distinguishes between two different roles: client and server.

In MQTT terms, a client is a program or device that uses MQTT. It always establishes the network connection to the server. A client can:

– publish application messages that other clients might be interested in;

– subscribe to request application messages that it is interested in receiving;

– unsubscribe to remove a request for application messages;

– disconnect from the server.

An MQTT server is an entity that accepts connections from clients. Unlike HTTP, it generally does not run any application logic, instead an MQTT server acts as an intermediary between clients publishing application messages and the clients that have subscribed to receive them.

The MQTT specification [OASIS MQTT] recommends the use of IANA-registered ports 1883 (MQTT over raw TCP/IP) and 8883 (MQTT running over TLS).

Although the MQTT protocol is relatively simple to implement, applications normally make use of pre-built implementations as follows.

– The applications themselves link to libraries that provide the MQTT client functionality. Libraries are available for a variety of programming languages and operating environments.

– The MQTT server functionality can be provided by a standalone software process (possibly running on a server that is remote from the clients), a hardware appliance or a cloud-hosted MQTT service.

The Eclipse foundation through their M2M working group, provides open source MQTT client code via its Paho project, and an open source server implementation via its Mosquitto project. Other open source and commercial implementations are also available.

I.3 MQTT details

I.3.1 Addressing a message – Topics and subscriptions

The MQTT protocol is based on the principle of publishing messages or subscribing to topics, or "pub/sub". See Figure I.3.1-1. Multiple clients connect to an MQTT server and subscribe to topics that they are interested in by sending an MQTT request protocol packet to the server. Clients also connect to the server and publish messages to the server, each message being associated with a topic. Many clients can subscribe to the same topics. The combination of the MQTT protocol and its server provides a simple, common interface for clients to connect to. A publisher can publish a message once and it be received by multiple subscribers.

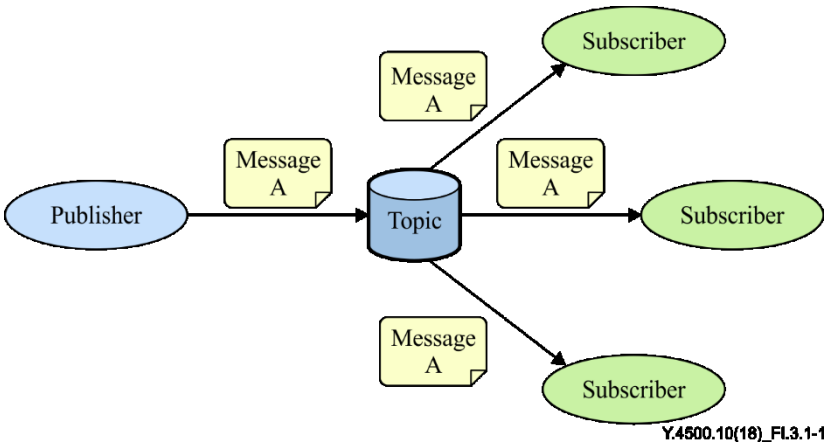

Figure I.3.1-1 – MQTT publish or subscribe messaging

A message in MQTT is associated with a topic when it is published. 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 it is 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 client registers its interest in topics by providing one or more topics filters. A topic filter can be a simple topic name or it can contain special "wildcard" characters that allow clients to subscribe to multiple topics at once, within a single level or within multiple levels in a topic tree.

I.3.2 Reliability

MQTT defines three levels of quality of service (QoS). The QoS defines how hard the server and client will try to ensure that a message is received. Messages can be sent at any QoS level and this affects the way the message is transmitted from the client to the server. When a client requests a subscription, it requests the maximum QoS at which it wants to receive messages on that subscription. This controls the way that messages matching that subscription are transmitted from the server to that client. The QoS used to transmit a message from the server is always less than or equal to the QoS used to transmit it to the server. 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.

QoS 0 messages are the least reliable. They are sent from client to server (or server to client) with no acknowledgement flowing in the opposite direction. A server is free to discard such messages.

QoS 1 is intended for idempotent messages. These messages are transmitted with a short packet ID. When a client (or server) receives such a message, it sends an acknowledgement packet back to the message sender. The sender is required to save a copy of that message until it receives the acknowledgement, and if there is a loss of network connectivity before it receives that acknowledgement, it is required to resend the message when connectivity is restored.

QoS 2 provides exactly once delivery. It uses a two-step acknowledgement protocol, in which both steps can be repeated an arbitrary number of times (if there is a loss of connectivity) without causing duplication of the original application message. Both client and server are required to save a copy of the message during this process.

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

In order to be able to continue with the QoS 1 or QoS 2 delivery protocols after a network reconnection, the server needs to have a way of distinguishing the individual clients that connect to it. It does this by means of an identifier called a Client ID. A client provides this ID when it first connects and the server records it and uses it as a key to any server-side state (such as the status of incomplete message delivery) associated with that client. When the client reconnects it presents the same ID, and that allows message delivery to complete. The client ID in effect represents the ID of the MQTT session that is maintained between the client and the server.

I.3.3 Retained messages

When a client publishes a message it can request that the message be retained. This means that the server 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. At most one such message is retained for any single topic. 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 might have to wait a long time to receive an update. With a retained message, the client will receive an instant update.

Bibliography

[b-ITU-T Y.4500.11] Recommendation ITU-T Y.4500.11 (2018), *oneM2M – Common terminology*.

[b-ATIS.oneM2M.TS0010V241 ATIS.oneM2M.TS0010V241-2016, *MQTT protocol binding*.

[b-CCSA M2M-TS-0010] CCSA M2M-TS-0010-V2.4.1 (2016), *oneM2M Technical Specification – MQTT protocol binding*.

[b-ETSI TS 118 110] ETSI TS 118 110 V2.4.1 (2016), *oneM2M; MQTT Protocol Binding (oneM2M TS-0010 version 2.4.1 Release 2)*.

[b-TSDSI STD T1.oneM2M TS-0010] TSDSI STD T1.oneM2M TS-0010-2.4.1 V1.0.0 (2017), *MQTT protocol binding*.

[b-TTAT TS-M2M-0010] TTAT TS-M2M-0010v2.4.1 v1.5.1 (2016), *oneM2M – MQTT Protocol Binding v1.5.1*.

[b-TTC TS-M2M-0010] TTC TS-M2M-0010v2.4.1 (2016), *oneM2M Technical Specification – MQTT protocol binding*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2018