# ESR Consortium
# LWM2M-MQTT-1.0

*LWM2M over MQTT*
*Profile Specification*

*ESR030*

Reference: ESR-SPE-030-LWM2M-MQTT
Version: 1.0
Rev: ADraft8

**DEFINITIONS**

"ESR" means the Specification, including any modifications and upgrades, where these terms have been stated or referred to, and made available to You by ESR Consortium, including without limitation, texts, drawing, codes and examples.

"ESR Consortium" means the non-profit entity, registered in France in accordance with the French law of 1901.

"You" means the legal entity or entities represented by the individual executing this Agreement.

**READ RIGHTS**

Subject to the terms and conditions contained herein, ESR Consortium grants to You a non-exclusive, non-transferable, worldwide, and royalty-free license to view and read the ESR solely for purposes of Your internal evaluation.

**GENERAL TERMS**

THIS DOCUMENTATION IS PROVIDED "AS IS", WITHOUT WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED.

THE READING OF THE ESR AND ALL CONSEQUENCES ARISING THEREOF IS YOUR SOLE RESPONSIBILITY. ESR CONSORTIUM SHALL NOT BE LIABLE TO YOU FOR ANY LOSS OR DAMAGE CAUSED BY, ARISING FROM, DIRECTLY OR INDIRECTLY, OR IN CONNECTION WITH THE ESR.

**COPYRIGHT**

ESR Consortium does claim any right in this ESR. You are free to use this ESR to make any clean room implementations or derivative work as long as You don't claim that Your work is compliant with the ESR. Compliance tests are available from the ESR Consortium.

**MISCELLANEOUS**

This Agreement shall be governed by, and interpreted in accordance with French Law. In no event shall this Agreement be construed against the drafter.

This Agreement contains the entire understanding between the parties concerning its subject matter and supersedes any other agreement or understanding, whether written or oral, which may exist or have existed between the parties on the subject matter hereof.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION.

ESR CONSORTIUM MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN ANY ESR PUBLICATION AT ANY TIME.

# Contents

# Tables

# Illustrations

# 1 PREFACE TO LWM2M-MQTT 1.0 PROFILE, ESR030

This document defines the, *LWM2M-MQTT 1.0* profile.

## 1.1 Who Should Use this Specification

This specification targets the following audiences:

- Server architects who want to build a server implementation that complies to the LWM2M-MQTT profile specification;
- Device architects who want to build an embedded client implementation that complies to the LWM2M-MQTT profile specification;

## 1.2 How This Specification is Organized

This specification is organized as follow:

- **Introduction** is a short chapter explaining what is LWM2M-MQTT, why it has been designed and what are its main assets.
- **Specification** is the chapter explaining in details the LWM2M-MQTT concepts and semantic.
- **Appendixes** is a chapter with additional content that helps implementors understanding the LWM2M-MQTT specification.

## 1.3 Comments

Your comments about LWM2M-MQTT are welcome. Please send them by electronic mail to the following address: `comments @e-s-r.net`, with LWM2M-MQTT in your subject line.

## 1.4 Glossary

- *ESR*: Embedded Specification Request
- *MQTT*: Message Queuing Telemetry Transport
- *LWM2M*: Lightweight Machine to Machine
- *CoAP*: Contrained Application Protocol

## 1.5 Document Conventions

In this document, references to pieces of code, standardized tokens, error codes … are written using the default monospace font (e.g. `PUT)`.

## 1.6 Implementation Notes

The LWM2M-MQTT specification does not include any implementation details. LWM2M-MQTT implementors are free to use whatever techniques they deem appropriate to implement the specification.

# 2 DOCUMENTS AND DEFINITIONS

## 2.1 References

*[CoAP]:* Shelby, Z., Hartke, K., Bormann, C., and B. Frank, “Constrained Application Protocol (CoAP)”, https://tools.ietf.org/html/rfc7252, June 2014.

*[LWM2M]:* Open Mobile Alliance, “Lightweight Machine to Machine Technical Specification”, Candidate Version 1.0, 07 April 2016

*[MQTT]:* Oasis, “MQTT Version 3.1.1 Plus Errata 01”, OASIS Standard Incorporating Approved Errata 01, 10 December 2015

*[BLOCK]:* Bormann, C. and Shelby, Z, "Blockwise transfers in CoAP", Work in Progress, October 2013.

*[CORE]:* Shelby, Z, “Constrained RESTful Environments (CoRE) Link Format”, https://tools.ietf.org/html/rfc6690, August 2012

*[OBSERVE]:* Hartke, K. “Observing Resources in CoAP”, https://tools.ietf.org/html/rfc7641, September 2015

## 2.2 Definitions

## 2.3 [MQTT] Abstract

*MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.*

*The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. Its features include:*

- *Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.*
- *A messaging transport that is agnostic to the content of the payload.*
- *Three qualities of service for message delivery:*
  - *"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 messages are assured to arrive but duplicates can occur.*
  - *"Exactly once", where message are assured to arrive 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 exchanges minimized to reduce network traffic.*
- *A mechanism to notify interested parties when an abnormal disconnection occurs.*

## 2.4 [LWM2M] Introduction

*This enabler defines the application layer communication protocol between a LWM2M Server and a LWM2M Client, which is located in a LWM2M Device. The OMA Lightweight M2M enabler includes device management and service enablement for LWM2M Devices. The target LWM2M Devices for this enabler are mainly resource constrained devices. Therefore, this enabler makes use of a light and compact protocol as well as an efficient resource data model.*

*A Client-Server architecture is introduced for the LWM2M Enabler, where the LWM2M Device acts as a LWM2M Client and the M2M service, platform or application acts as the LWM2M Server. The LWM2M Enabler has two components, LWM2M Server and LWM2M Client. Four interfaces are designed between these two components as shown below:*

- *Bootstrap*
- *Client Registration*
- *Device management and service enablement*
- *Information Reporting*

*[...] The LWM2M Enabler uses the Constrained Application Protocol (CoAP) with UDP and SMS bindings. Datagram Transport Layer Security (DTLS) provides security for UDP transport layer.*

# 3 INTRODUCTION

The LWM2M protocol is becoming the standard for remotely managing a device through a set of normalized operations. However, although its semantics can be implemented on any kind of protocols, the LWM2M specification is written in a way that it is tightly coupled to the CoAP protocol, compliant with a lossy network such as UDP or SMS transport.

On another hand, most of the IoT devices are connected to a central server using a regular TCP connection, for two main reasons:

- This is no more an issue from server side to keep billions of TCP connections up at a time.
- Although the device is viewed as a server that exposes resources upon which requests are initiated from the cloud, the connection is initiated by the device, which removes trying to resolve NAT traversal issues.

There are many initiatives for proposing an implementation of CoAP over TCP, thus having LWM2M implementations already running on a TCP transport without any effort.

The MQTT broker is one of the most commonly used central server, especially because its publish/subscribe paradigm eases sending data to the cloud.

The goal of this specification is to propose an alternative composition of LWM2M semantics with CoAP message format based on MQTT publish/subscribe paradigm on top of a TCP transport. The key points driving choices made in this specification are:

- Keeping a single open link from device to the cloud for data and device management
- Minimizing embedded client footprint for targeting resource-constrained devices, such as those running on microcontrollers or small embedded microprocessors. Especially trying to avoid as much as possible mechanisms specified by CoAP to manage unordered or loss packets (binary data streaming instead of unordered blockwise transfer, de-duplication, sending again after timeouts...), as it is based on a TCP transport that implies no packets loss

- Relying as much as possible on the original specifications

This specification assumes the reader refers to *[LWM2M]*, *[CoAP]*, *[MQTT]* specifications and only describes differences and modifications (subset / updates) compared to the original specifications.

# 4 SPECIFICATION

This specification defines the use of CoAP messages transferred as payload of general-purpose MQTT topics. MQTT is just viewed as a high-level transport layer for CoAP messages.

A LWM2M Client and a LWM2M server are connected through two general-purpose MQTT topics named *transport topics* that serve for bi-directional communications (see section 4.1.1 for naming convention). Each pair of transport topics connects a single LWM2M Client and a single LWM2M server. As specified by *[LWM2M]*, a LWM2M client may target multiple LWM2M servers through the same MQTT broker connection using distinct transport topics pairs (see Illustration 4-1 for transport topics pairs examples).

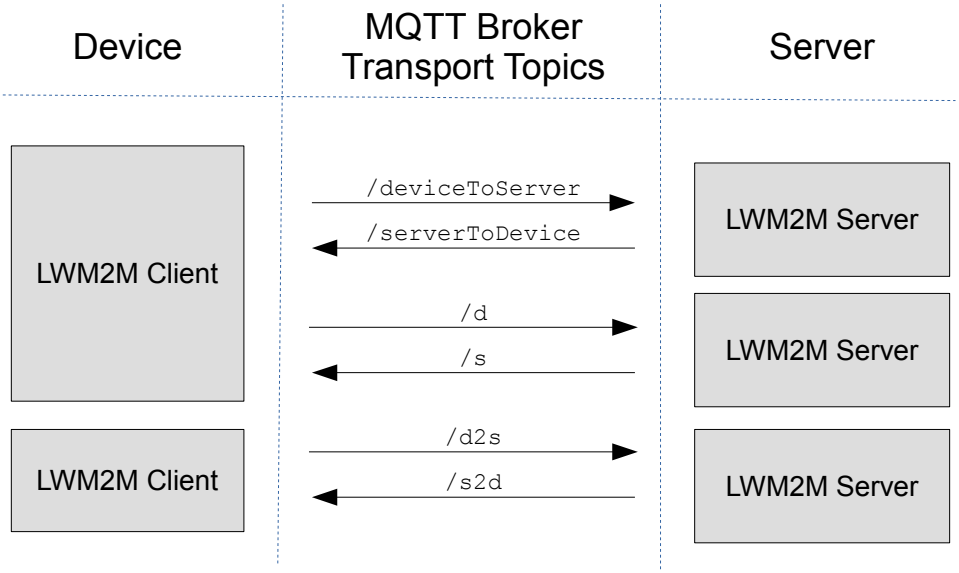


*Illustration 4-1: LWM2M over MQTT Overview*

## 4.1 Transport Topics

### 4.1.1 Naming Convention

Transport topics names are structured as follows:

```
[PREFIX]/[DEVICE_ID]/[DIRECTION]
```

Where:

- `[PREFIX]` is an implementation-specific topic prefix (can be empty)
- `[DEVICE_ID]` is a unique device identifier under the chosen `[PREFIX]`

- `[DIRECTION]` is the topic name indicating the message direction. By default, the topic name for communications (i.e. publication) from server to device (resp. device to server) is `serverToDevice` (resp. `deviceToServer`). Topics names may be set in the Security Object during the bootstrap sequence (see 4.5.2).

Example of valid topics:

```
mymqttbroker/04ac-a451-ff43/deviceToServer
0/deviceToServer
mybroker/myaccount/0001/serverToDevice
```

### 4.1.2 Topic Payloads

Every message published as payload on an MQTT transport topic represents exactly one CoAP message.

### 4.1.3 Access Restriction

The MQTT broker MUST provide a mean to restrict use of these topics to device it belongs.

## 4.2 MQTT Settings

This specification assumes the LWM2M Client and the LWM2M server are connected to the MQTT broker. When the network connection of one of the two parts is broken the messages exchanged are lost. No session state is maintained across multiple network connections/disconnections.

### 4.2.1 Specification Version

The MQTT broker MUST implement MQTT version 3.1.1 (*[MQTT]*) or higher.

### 4.2.2 Quality Of Service

Publication of and subscription to transport topics MUST be done with MQTT quality of service level 0.

### 4.2.3 Clean Session

LWM2M Client and LWM2M server MUST connect to the MQTT broker with the `CleanSession` flag set to 1.

### 4.2.4 Retain

All publications to transport topics MUST be done with the `RETAIN` flag set to 0.

## 4.3 Messages Sequences

The following sequence diagram shows the packet sequences transferred on transport topics illustrated with the *[LWM2M]* section *5.3 Client Registration Interface*.

Basically, the initiator (the device in this case) publishes a first CoAP request. Then the MQTT broker distributes the message to the subscriber(the server in this case). When the request has been processed, the server sends the response to the device (in this case the resource has been created from server side means that the device has been successfully registered).

Note that in order to receive device requests, the server first has to subscribe to each `deviceToServer` topic it may be connected to. In this example, this is done by subscribing to any device using a single level wildcard (+).

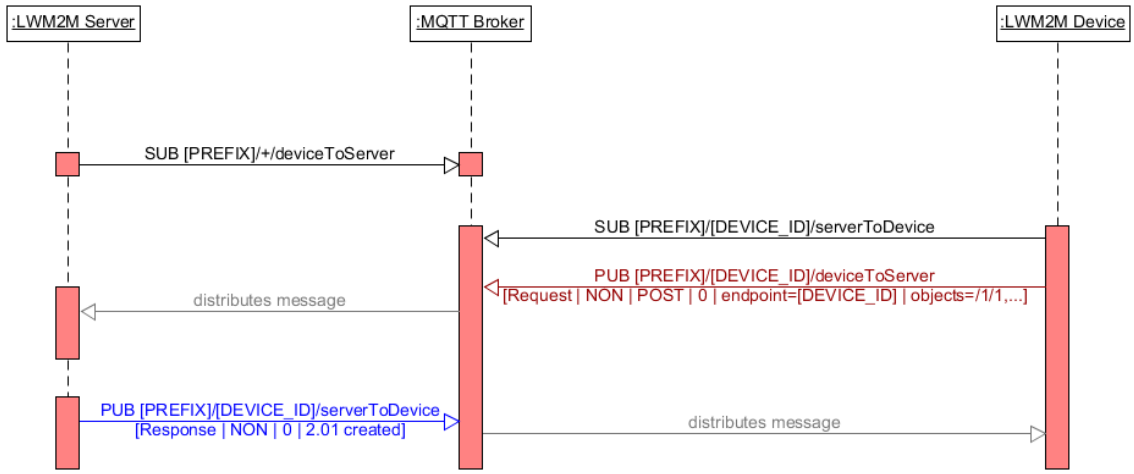


*Illustration 4-2: Message Packets Sequence for LWM2M Client Registration Interface*

## 4.4 Published Messages Rules

### 4.4.1 Message Format

Table 4-1 specifies the different kind of message types that are allowed to be transferred through MQTT topics.

| Message Type | Mnemonic | Allowed | Comment |
|---|---|---|---|
| Acknowledgment | ACK | NO | Acknowledgments are only used in return to Confirmable messages, which are not allowed. |
| Confirmable | CON | NO | This specification removes all CoAP mechanisms related to reliabilty. |
| Non-confirmable | NON | YES | CoAP messages requests and responses MUST be marked with type Non-confirmable (NON). |
| Reset | RST | YES | Reset messages are only allowed in case of observe pattern to cancel a notification (see section 5.1). |

*Table 4-1: CoAP Allowed Message Types*

### 4.4.2 Reliability

The [*CoAP*] section *4.2. Messages Transmitted Reliably* is overridden by this specification. When a CoAP request timeout is reached, the message is not retransmitted. The pending request is dropped and an error shall be returned to the request initiator.

The *[CoAP]* section *4.5. Message Deduplication* is overridden by this specification. According to the MQTT QoS 0 behavior, a message cannot be received more than once.

### 4.4.3 Transmission Parameters

This specification overrides the *[COAP]* section *4.8. Transmission Parameters*.

A request is considered to be expired if no response is received after `REQUEST_TIMEOUT` delay.

```
+-------------------+---------------+
| name              | default value |
+-------------------+---------------+
| REQUEST_TIMEOUT   | 15 seconds    |
+-------------------+---------------+
```

A received CoAP response to an unknown request (i.e. a request with a token that does not match any of the previously sent and non expired requests) MUST be ignored.

## 4.5 LWM2M Extensions

This section describes LWM2M over MQTT extensions to *[LWM2M]* specification.

### 4.5.1 Binding Mode

This specification defines an additional binding mode (called MQTT) to the list described in *[LWM2M]* section *5.3.1.1 Behavior with Current Transport Binding and Mode*.

| Current Transport Binding and Mode | Behaviour |
|---|---|
| M (MQTT) | The LWM2M Server expects that the LWM2M Client is reachable via the MQTT binding at any time. The LWM2M Server MUST send requests to a LWM2M Client using the MQTT binding. The LWM2M Client MUST send the response to such a request over the MQTT binding. |

*Table 4-2: New MQTT Binding and Mode*

### 4.5.2 Bootstrap Security Object

When a device first connects a LWM2M Bootstrap Server, one ore more LWM2M Security Object is written on the device. Table 4-3 extends the Security Object defined in *[LWM2M]* section *E.1. LWM2M Object: LWM2M Security*. When a resource ID already exists, it overrides the previous definition.

| ID | Name | Instances | Mandatory | Type | Range or Enumeration | Description |
|---|---|---|---|---|---|---|
| 0 | LWM2M Server URI bytes | Single | Mandatory | String | 0-255 | Uniquely identifies the LWM2M Server or LWM2M Bootstrap Server over MQTT, and is in the form of a MQTT URI: `tcp://host:port` or `ssl://host:port`, where host is an IP address or FQDN, and port is the TCP port of the Server. |
| 100 | Topic Prefix | Single | Optional | String | 0-255 | MQTT transport topics prefix (default value means there is no prefix). See section 4.1.1. |
| 101 | Server To Device Topic Name | Single | Optional | String | 0-255 | MQTT topic simple name for communication from server to device (default value is `serverToDevice` – see section 4.1.1). |
| 102 | Device To Server Topic Name | Single | Optional | String | 0-255 | MQTT topic simple name for communication from device to server (default value is `serverToDevice` – see section 4.1.1). |

*Table 4-3: LWM2M Security Object Extensions*

## 4.6 Binary Data Transfer

### 4.6.1 Introduction

Although it is not clearly specified in *[LWM2M]* specification, implementations often rely on the CoAP Blockwise Transfer Layer specification *([BLOCK])* for transferring large data in LWM2M Read or Write resources operations (especially when the resource type is opaque).

This specification DOES NOT support the CoAP Blockwise Transfer layer specification *[BLOCK]* for LWM2M operations.

All LWM2M operations MUST be performed by publishing a single CoAP message on a MQTT topic[1].

In addition, this specification introduces a new protocol dedicated for large binary transfer through MQTT topics without having to open another connection. It can be used to any resource holding an URI link such as the *PackageURI* resource of the *[LWM2M]* Firmware Update object. For this purpose, a new URI scheme is defined.

The binary data can be retrieved by blocks, giving the receiver a full control of the data to be received:

- It decides to send a request to get a new block when it is ready

1 The theoretical maximum size of a CoAP message is limited by the MQTT maximum payload size

- It decides the size of the block to be received, depending on its memory capabilitites
- It decides the order the blocks are received, allowing to process the resource in a streaming mode
- It can request the same block multiple times

### 4.6.2 Block Option

Binary data transfer protocol is based on the `Block1` and `Block2` options of the *[BLOCK]* specification (sections *2.1. The Block2 and Block1 Options* and *2.2. Structure of a Block Option*). The Block option is extended to a variable size of up to 4 bytes allowing to transfer binary chunks greater than 1KB:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   NUM                 |M|       SZX           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

### 4.6.3 URI Scheme

Binary data transfer protocol defines the following URI:

```
mq2m://[owner]/[path]?size=xxx
```

where:

- `owner` is the owner of the resource, either `server` or `device`.
- `path` is a local path to the data in the owner side
- `size` is the total size in bytes of the data

### 4.6.4 Protocol

- The data blocks can be retrieved by sending `GET` requests on the topic associated to the owner: `deviceToServer` (resp. `serverToDevice`) if the owner is `server` (resp. `device`), with the following options:
- `Block2`: the relative number 0 (`NUM`) and the desired size (`SZX`) of the packet.
- `URI-Path`: each segment of the owner local path of the data to be retrieved

Then the owner sends the data in a response with `2.05 Content` code and the following option:

- `Block1`: filled with the same block number, size and more blocks following flag `(M)`

Illustration 4-3 describes a typical sequence from transferring a firmware using this protocol.

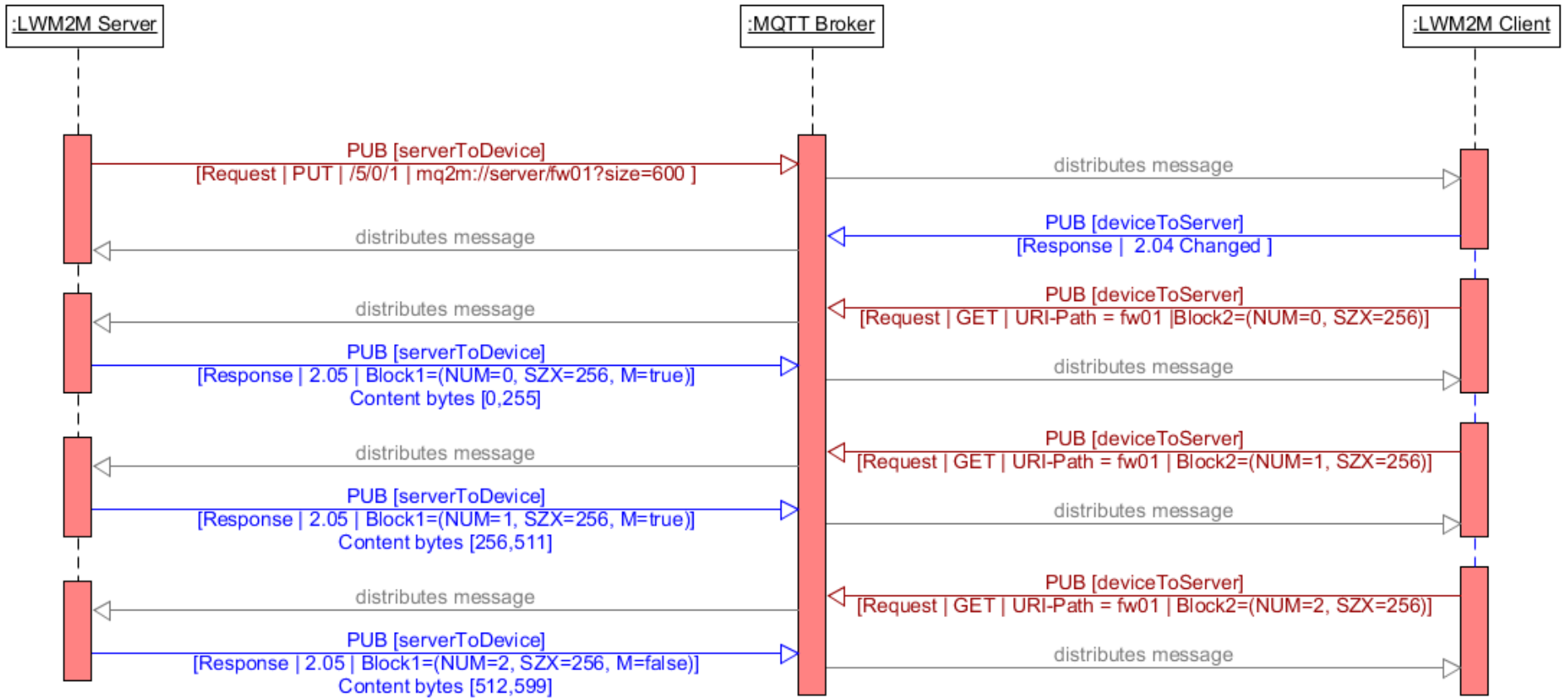


*Illustration 4-3: Example of Message Sequences for a LWM2M Firmware Update using the Binary Data Transfer Protocol*

# 5 APPENDIXES

## 5.1 CoAP Specification Mapping

Table 5-1 lists the *[CoAP]* specification sections that are included by this specification without any modifications. When a section is in the list, its potential subsections are also included.

| |
|---|
| 2.1. Messaging Model |
| 2.2. Request/Response Model |
| 3.1. Option Format |
| 3.2. Option Value Formats |
| 4.1. Messages and Endpoints |
| 4.4. Message Correlation |
| 5.1. Requests |
| 5.2. Responses |
| 5.2.2. Separate |
| 5.3. Request/Response Matching |
| 5.4. Options |
| 5.5. Payloads and Representations |
| 5.8. Method Definitions |
| 5.10. Option Definitions |
| 12. IANA Considerations |

*Table 5-1: CoAP Included Sections*

Table 5-2 lists the *[CoAP]* specification sections that are overridden by this specification.

| Section | Comment |
|---|---|
| 3 Message | |
| 4.2. Messages Transmitted Reliably | This specification removes all CoAP re-transmission mechanisms and messages deduplication. See section 4.4.2. |
| 4.3. Messages Transmitted without Reliability | |
| 4.5. Message Deduplication | |
| 4.6. Message Size | This specification does not limit the size of a CoAP message, which was originally limited to the maximum size of an UDP packet. |
| 4.8. Transmission Parameters | As the transmission link is reliable and message exchange is fully controlled by both parts, all timeouts related to message retransmission have been removed, and only an applicative request processing timeout has been introduced. See section 4.4.3. |
| 5.2.3. Non-confirmable | This specification enforces that requests and responses messages are always marked non-confirmable. See section 4.4. |

*Table 5-2: CoAP Overridden Sections*

Table 5-3 lists the *[CoAP]* specification sections that are not included by this specification. Some of them are not included because they are not required by the *[LWM2M]* specification. See also *[LWM2M]* section *8.1 Required Features*.

| Section | Comment |
|---|---|
| 2.3. Intermediaries and Caching | This section is an introduction. See related comments on *[5.6. Caching]*, *[5.7. Proxying]* and *[10. Cross-Protocol Proxying between CoAP and HTTP]*. |
| 2.4. Resource Discovery | This section is an introduction. See related comments on *[7. Discovery]*. |
| 4.7. Congestion Control | As the transmission link is reliable and message exchange is fully controlled by both parts, there is no need to handle outstanding interactions cases. See also *[4.2. Messages Transmitted Reliably]* and section 4.4.3 defining acknowledgment and response timeouts. |
| 5.2.1. Piggybacked | This specification removes acknowledgment messages types. See section 4.4.1. |
| 5.6. Caching | This is not required by *[LWM2M]* specification. |
| 5.7. Proxying | This is not required by *[LWM2M]* specification. |
| 7. Discovery | This is not required by *[LWM2M]* specification. See also *[LWM2M]* section *5.3 Client Registration Interface.* |
| 8. Multicast CoAP | This is not required by *[LWM2M]* specification and this specification deprecates it as it relies on a TCP connection. |
| 9. Securing CoAP | This specification deprecates DTLS binding for CoAP it as it defines a new binding based on a TCP connection. |
| 10. Cross-Protocol Proxying between CoAP and HTTP | This is not required by *[LWM2M]* specification. |

*Table 5-3: CoAP Not Included Section*

Table 5-4 lists the *[CoAP]* specification sections that are out of scope of this specification (not related to the protocol).

| Section | Comment |
|---|---|
| 6. CoAP URIs | The URI mapping of CoAP resources is not involved in the CoAP message format. |
| 11. Security Considerations | For securing the MQTT connection, refer to non normative security section of *[MQTT]*. |

*Table 5-4: CoAP Out of Scope Sections*

## 5.2 Observe Specification Mapping

Table 5-5 lists the *[OBSERVE]* specification sections that are included by this specification without any modifications. When a section is in the list, its potential subsections are also included.

| |
|---|
| 2. The Observe Option |
| 3.1. Request |
| 3.2. Notifications |
| 3.4. Reordering |
| 3.6. Cancellation |
| 4.1. Request |
| 4.2. Notifications |
| 4.4. Reordering |

*Table 5-5: Observe Included Sections*

Table 5-6 lists the *[OBSERVE]* specification sections that are overridden by this specification.

| Section | Comment |
|---|---|
| 3.5. Transmission | Requests and Responses (i.e. notifications) MUST be sent as non-confirmable messages. See section 4.4. Other features mentioned in this section are left as-is. |
| 4.5. Transmission | |
| 4.5.1. Congestion Control | This specification removes all CoAP re-transmission mechanisms and messages deduplication. See section 4.4.2. |
| 4.5.2. Advanced Transmission | |

*Table 5-6: Observe Overridden Sections*

Table 5-7 lists the *[OBSERVE]* specification sections that are not included by this specification. Some of them are not included because they are not required by the *[LWM2M]* specification. See also *[LWM2M]* section *8.1 Required Features*.

| Section | Comment |
|---|---|
| 3.3. Caching | This is not required by *[LWM2M]* specification. |
| 4.3. Caching | This is not required by *[LWM2M]* specification. |
| 5. Intermediaries | This is not required by *[LWM2M]* specification. |

*Table 5-7: Observe Not Included Section*

Table 5-8 lists the *[OBSERVE]* specification sections that are out of scope of this specification (not related to the protocol).

| Section | Comment |
|---|---|
| 6. Web Linking | The web link mapping of an observed CoAP resource is not involved in the CoAP message format. |
| 7. Security Considerations | For securing the MQTT connection, refer to non normative security section of *[MQTT]*. |

*Table 5-8: Observe Out of Scope Sections*