Lightweight Machine to Machine Technical Specification

Candidate Version 1.0 – 07 Apr 2016

Open Mobile Alliance

OMA-TS-LightweightM2M-V1_0-20160407-C

Contents

Figures

Tables

1. Scope

The present document specifies the LightweightM2M protocol and the core set of LightweightM2M Objects.

2. References

2.1 Normative References

[3GPP-TS_23.003]	3GPP TS 23.003 “Numbering, addressing and identification”
[3GPP-TS_23.038]	3GPP TS 23.038 “Alphabets and language-specific information”
[3GPP-TS_23.040]	3GPP TS 23.040 “Technical realization of the Short Message Service (SMS)”
[3GPP-TS_31.111]	3GPP TS 31.111 “Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)”
[3GPP-TS_31.115]	3GPP TS 31.115 “Remote APDU Structure for (U)SIM Toolkit applications”
[CoAP]	Shelby, Z., Hartke, K., Bormann, C., and B. Frank, “The Constrained Application Protocol (CoAP)” IETF RFC 7252 – June 2014
[CoRE_Interface]	Z. Shelby, M. Vial, “CoRE Interfaces”, draft-ietf-core-interfaces-01, Nov 2013
[ETSI TS 102.221]	“Smart Cards; UICC-Terminal interface; Physical and logical characteristics”, (ETSI TS 102 221 release 11), URL:http://www.etsi.org/
[ETSI TS 102.223]	“Smart Cards; Card Applications Toolkit (CAT) (Release 11)” URL:http://www.etsi.org/
[ETSI TS 102.225]	ETSI TS 102 225 (V11.0.0): “Smart Cards; Secured packet structure for UICC based applications (Release 11)” URL:http://www.etsi.org/
[FLOAT]	IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008
[GLOBALPLATFORM]	GlobalPlatform v2.2.1 - January 2011 -
[GP SCP03]	GlobalPlatform Secure Channel Protocol 03 (SCP 03) Amendment D v1.1 Sept 2009
[IEEE 754-2008]	IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008
[IOPPROC]	“OMA Interoperability Policy and Process”, Version 1.1, Open Mobile Alliance™, OMA-IOP-Process-V1_1, URL:http://www.openmobilealliance.org/
[LWM2M-AD]	“Lightweight Machine to Machine Architecture”, Open Mobile Alliance™, OMA-AD-LightweightM2M-V1_0, URL:http://www.openmobilealliance.org/
[OBSERVE]	Hartke, K. “Observing Resources in CoAP”, draft-ietf-core-observe-10 (work in progress), September 2013.
[PKCS#15]	“PKCS #15 v1.1: Cryptographic Token Information Syntax Standard”, RSA Laboratories, June 6, 2000. URL:ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf
[RFC2119]	“Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[RFC2234]	“Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt
[RFC4122]	“A Universally Unique Identifier (UUID) URN Namespace”, P. Leach, et al. July 2005, URL:http://www.ietf.org/rfc/rfc4122.txt
[RFC5246]	The Transport Layer Security (TLS) Protocol Version 1.2
[RFC5289]	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)

[RFC5487] Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for TLS", RFC6655, July 2012.

[RFC6690] Shelby, Z. "Constrained RESTful Environments (CoRE) Link Format", RFC6690, Aug 2012.

[SENML] C. Jennings, Z. Shelby, J. Arkko, "Media Types for Sensor Markup Language (SENML)", draft-jennings-senml-10 (work in progress), April 2013.

[TR-069] Broadband Forum: "TR-069 CPE WAN Management Protocol" Issue: 1 Amendment 5. URL:http://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf

[WAP-WDP] Wireless Application Protocol Forum, "Wireless Datagram Protocol", June 2001.

2.2 Informative References

[3GPP TS 31.116] 3GPP TS 31.116 (V10.2.0): "Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications (Release 10)"

[3GPP2 C.S0078-0] 3GPP2 C.S0078-0 (V1.0): "Secured packet structure for CDMA Card Application Toolkit (CCAT) applications"

[3GPP2 C.S0079-0] 3GPP2 C.S0079-0 (V1.0) "Remote APDU Structure for CDMA Card Application Toolkit (CCAT) applications"

[DMREPPRO] "OMA Device Management Representation Protocol, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_RepPro-V1_3. URL:http://www.openmobilealliance.org

[ETSI TS 102 226] ETSI TS 102 226 (V11.0.0): "Smart cards; Remote APDU structure for UICC based applications (Release 11)"

[OMADICT] "Dictionary for OMA Specifications", Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

[SMS-DTLS] Datagram Transport Layer Security (DTLS) over Global System for Mobile Communications (GSM) Short Message Service (SMS), URL:http://www.ietf.org/id/draft-fossati-dtls-over-gsm-sms-01.txt

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

LWM2M Bootstrap Server Account	LWM2M Security Object Instance with Bootstrap Server Resource true
LWM2M Server Account	LWM2M Security Object Instance with Bootstrap Server Resource false and associated LWM2M Server Object Instance
Non-persistent Key	Key that has low entropy e.g. because it is derived from a PIN or password or device serial number, or, derived from a global secret stored on multiple devices.
Persistent Key	Key that can with high probability be kept secret for the lifetime of the device. This requires that at minimum each key is unique per device, that each key has high entropy, and that each key retains high entropy even given knowledge of keys extracted from other devices.
Queue Mode	The interaction model between an LWM2M Client and LWM2M Server is based on that LWM2M Server queues the requests.

Kindly consult [OMADICT] for more definitions used in this document.

3.3 Abbreviations

LWM2M	Lightweight Machine to Machine (refers to this OMA enabler)

Kindly consult [OMADICT] for more abbreviations used in this document.

4. 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 LWNM2M 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

This architecture is shown in Figure 1. 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. The LWM2M Enabler protocol stack is shown in Figure 2.

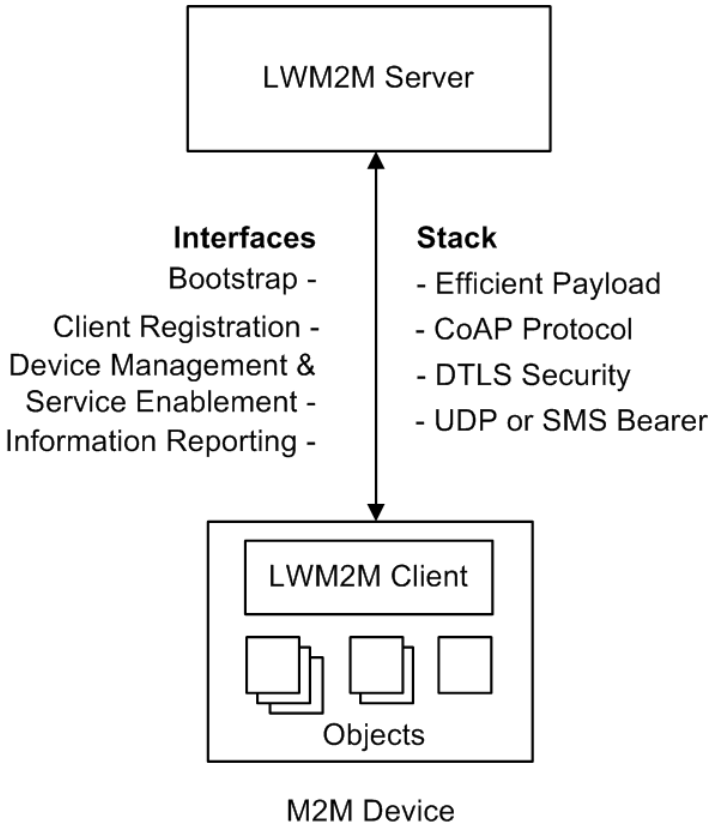


Figure 1: The overall architecture of the LWM2M Enabler.

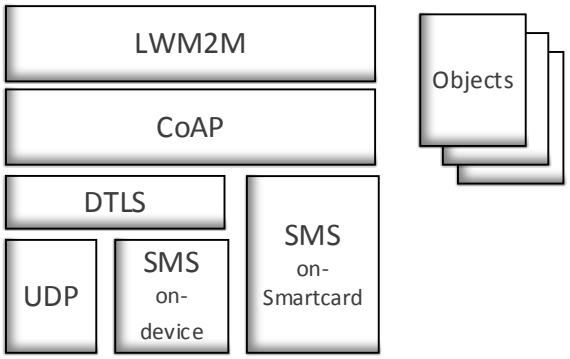


Figure 2: The protocol stack of the LWM2M Enabler.

4.1 Version 1.0

Lightweight M2M 1.0 enabler introduces the following features below for the initial release.

- Simple Object based resource model
- Resource operations of creation/retrieval/update/deletion/configuration of attribute
- Resource observation/notification
- TLV/JSON/Plain Text/Opaque data format support
- UDP and SMS transport layer support
- DTLS based security
- Queue mode for NAT/Firewall environment
- Multiple LWM2M Server support
- Basic M2M functionalities: LWM2M Server, Access Control, Device, Connectivity, Firmware Update, Location, Connectivity Statistics

5. Interfaces

According to the architecture diagram [LWM2M-AD], there are four interfaces: 1) Bootstrap, 2) Client Registration, 3) Device Management and Service Enablement, and 4) Information Reporting. The operations for the four interfaces can be classified as uplink operations and downlink operations. The operations of each interface are defined in this section, and then mapped to protocol mechanisms in Section 7 Transport Layer Bindings and Encodings.

Figure 3 shows the operation model for interface "Bootstrap". For this interface, the operations are uplink operation named "Request Bootstrap" and a downlink operation named "Write" and "Delete". These operations are used to initialize the needed Object(s) for the LWM2M Client to register with one or more LWM2M Servers. With the "Write" operation on this interface, the LWM2M Client MUST write the value included in the payload regardless of an existence of the targeting Object Instance(s) or Resource(s) and access rights. In the mode where the Server is addressing the Bootstrap Information to the LWM2M Client, the Server MUST inform the LWM2M Client when this transfer is over by sending a Bootstrap Finish command.

Bootstrapping is also defined using Factory Bootstrap (e.g. storage in Flash) or Bootstrap from Smartcard (storage in a Smartcard).

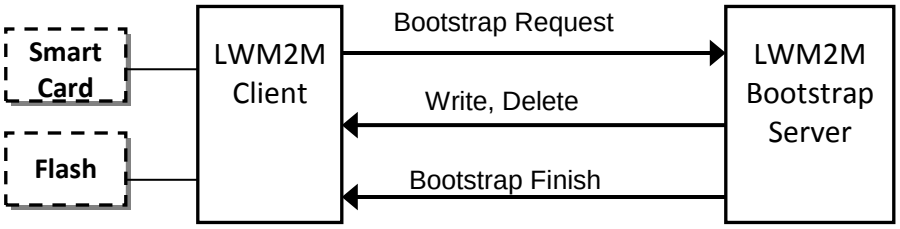


Figure 3: Bootstrap

Figure 4 shows the operation model for the interface "Client Registration". For this interface, the operations are uplink operations named "Registration", "Update" and "De-register".

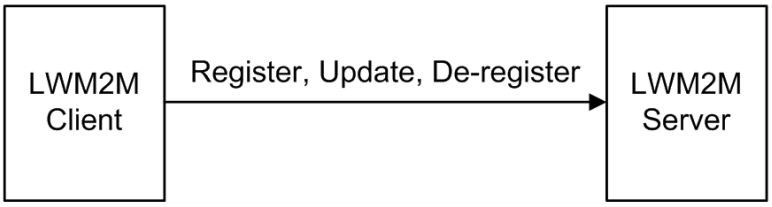


Figure 4: Client Registration

Figure 5 shows the logical operation model for interface "Device Management and Service Enablement". For this interface, the operations are downlink operations named "Read", "Create", "Delete", "Write", "Execute", "Write Attributes", and "Discover". These operations are used to interact with the Resources, Resource Instances, Objects, Object Instances and/or their attributes exposed bythe LWM2M Client. The "Read" operation is used to read the current values; the "Discover" operation is used to discover attributes and to discover which Resources are implemented in a certain Object; the "Write" operation is used to update the values; the "Write Attributes" operation is used to change attribute values and the "Execute" operation is used to initiate an action. The "Create" and "Delete" operations are use to create or delete Instances.

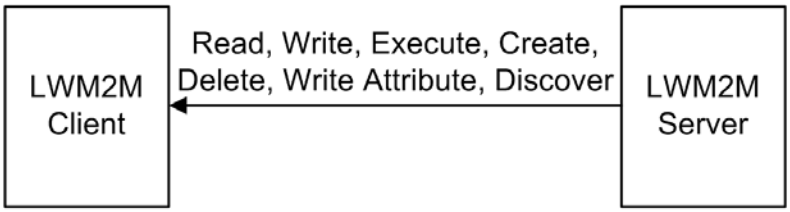


Figure 5: Device Management and Service Enablement

Figure 6 shows the operation model for interface “Information Reporting”. For this interface, the operations are downlink operations “Observe” or “Cancel Observation” and an uplink operation “Notify”. This interface is used to send the LWM2M Server a new value related to a Resource on the LWM2M Client.

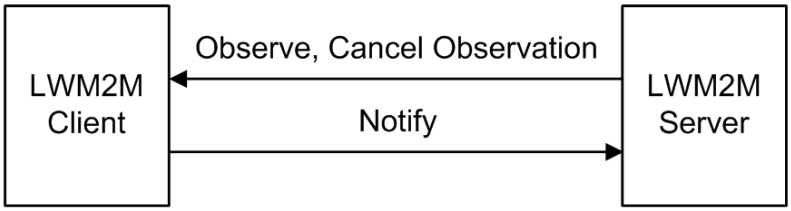


Figure 6: Information Reporting

The relationship between operations and interfaces is listed in the following Table 1.

Interface	**Direction**	**Operation**
Bootstrap	Uplink	Request Bootstrap
Bootstrap	Downlink	Write, Delete
Client Registration	Uplink	Register, Update, De-register
Device Management and Service Enablement	Downlink	Create, Read, Write, Delete, Execute, Write Attributes, Discover
Information Reporting	Downlink	Observe, Cancel Observation
Information Reporting	Uplink	Notify

Table 1: Relationship of operations and interfaces

5.1 Attributes

5.1.1 Attributes Definitions and Rules

Attributes are metadata which can be attached to an Object, an Object Instance or a Resource and may be different for each related LWM2M Server. These attributes can fulfill various roles: from just carrying information (e.g. Discover), up to containing parameters for Notification for example.

Attributes attached to Object, Object Instance, Resource, are respectively named O-Attribute, OI-Attribute, R-Attribute.

These Attributes MAY be carried in the message payload of Registration and Discover operations; they also MAY be updated - when writable - through the “Write Attributes” operation.

Regardless to what element it is attached, an Attribute can be specified at (or assigned to) various levels: Object, Object Instance, Resource levels. Additionally prevalence rules apply when the same Attribute specifies a value at different levels:

- An O-Attribute MAY only be assigned to the Object level.
- An OI-Attribute MAY be assigned to the Object Instance level, but also to the Object level.
  - Rule 1: When assigned to both levels, the value of the OI-Attribute specified at Object Instance level, will prevail.
  - Rule 2: When assigned to the Object level, the scope of the OI-Attribute value extends to all the Instances of that Object, as long as the Rule 1 is respected.
- An R-Attribute MAY be assigned to 3 different levels: the Resource level, the Object Instance level and the Object level.
  - Rule 3: When assigned to the Resource level, the value of an R-Attribute prevails for that Resource whatever a value for this R-Attribute is also specified at an upper level (Object or Object Instance level).
  - Rule 4: When assigned to an Object Instance level, the scope of an R-Attribute value extends to all the Resources of that Object Instance to which this R Attribute is attached as long as the Rule 3 is respected.
  - Rule 5: When assigned to the Object level, the scope of an R-Attribute value extends to all the resources of any Instance of that Object, to which this R-Attribute is attached as long as the Rule 2 is respected.

An attribute is fully determined by several characteristics which are listed in the table below:

Attribute characteristics	Description
Name	Attribute Name used to reference a specific Attribute in that Enabler (e.g. “Minimum Period”)
CoRE Link Param	the string used when this Attribute is transferred to CoAP as a CoRE link parameter (ex pmin)
Attachment	The Object, Object Instance or Resource, to which an Attribute applies
Assignation Level	The Level (Object, Object Instance, Resource) where the Attribute is specified.
Class	Attributes are organized according to their purpose; 2 Class of Attributes are supported in LWM2M TS 1.0 [NOTIFICATION] gather Attributes regarding Notify operations parameters [PROPERTIES] gather Attributes regarding general informations
Access Mode	R, W, RW: operation allowed by the LWM2M Server.
Applicability	Condition to fulfil for allowing to attach such an Attribute
Default Value	<value> | -
Value Type	LWM2M data type
Value	The Value carried by this Attribute : its data type must be of “Value Type”

Table 2: Attribute Characteristics

Some Attributes MAY be exposed to the LWM2M Server in the payload response of a “Discover” command (Section 5.4.2).

The value of some Attributes MAY be changed by the LWM2M Server in using the “Write Attributes” command (Section 5.4.4); which Attribute are concerned are marked as “W” (writable) in the table of the next Section.

Note: A LWM2M message payload is a list of application/link-format CoRE Links [RFC6690] which will includes the LWM2M Attributes

5.1.2 Attributes Classification

[PROPERTIES] Class Attributes

The role of these Attributes is to provide metadata which MAY communicate helpful information to LWM2M Server for example easing data management.

Except when specifically mentioned as required, the LWM2M Server and LWM2M Client SHOULD support [PROPERTIES] Class Attributes listed Table 3.

Attribute Name	CoRE Link param	Attachment	Assignation Level	Required	Access Mode	Value Type	Default Value	Applicability	Notes
Dimension	dim	Resource	Resource Level	YES (Client)	R	Integer [0:255]	-	Multiple Resource	Number of Instantiations for a Multiple Resource

Table 3: [PROPERTIES] Class Attributes

[NOTIFICATION] Class Attributes

The role of these R-Attributes is to provide parameters to the "Notify" operation; any readable Resource can have such R-attributes.

In the message sent by a LWM2M Client in response to an "Observe" operation, the current Resource value is reported; this event can be considered as the initial notification.

Each time a Resource notification is sent, the "Minimum Period" and "Maximum Period" timers associated to this Resource are restarted.

The notification of a Resource value will be sent when the combination of a change value condition ("Greater Than", "Less Than", "Step) and the "Minimun Period" timing conditions are both fulfilled for that Resource.

The LWM2M Server MUST support and LWM2M Client SHOULD support all the [NOTIFICATION] Class Attributes listed Table 4

Attribute Name	CoRE Link param	Attachment	Assignation Level	Required	Access Mode	Value Type	Default Value	Apply Condition
Minimum Period	pmin	Resource	Resource Level Object Instance Level Object Level	No	RW	Integer	0 (sec)	Readable Resource
Notes: The Minimum Period Attribute indicates the minimum time in seconds the LWM2M Client MUST wait between two notifications. If a Resource value has to be notified during the specified quiet period, the notification MUST be sent as soon as this period expires. In the absence of this parameter, the Minimum Period is defined by the Default Minimum Period set in the LWM2M Server Account.								
Maximum Period	pmax	Resource	Resource Level Object Instance Level Object Level	No	RW	Integer	-	Readable Resource
Notes: The Maximum Period Attribute indicates the maximum time in seconds the LWM2M Client MAY wait between two notifications. When this "Maximun Period" expires after the last notification, a new notification MUST be sent. In the absence of this parameter, the "Maximum Period" is defined by the Default Maximum Period set in the LWM2M Server Account. The maximum period parameter MUST be not smaller than the minimum period parameter.								
Greater Than	gt	Resource	Resource Level	No	RW	Float	-	Numerical &Readable Resource
Notes: This "Greater Than" Attribute defines a threshold high value. When this Attributes is present, the LWM2M Client MUST notify the Server each time the Observed Resource value crosses the "Greater Than" Attribute value with respect to pmin parameter.								
Less Than	lt	Resource	Resource Level	No	RW	Float	-	Numerical &Readable Resource

Notes: This "Less Than" Attribute defines a threshold low value. When this Attributes is present, the LWM2M Client MUST notify the Server each time the Observed Resource value crosses the "Less Than" Attribute value with respect to pmin parameter.								
Step	stp	Resource	Resource Level	No	RW	Float	-	Numerical &Readable Resource
Notes: This "Step" Attribute defines a minimum change value between two notifications. When this Attribute is present, the change value condition will occur when the value variation since the last notification of the Observed Resource, is greater or equal to the "Step" Attribute value. When the "Step" change value condition occurs, the LWM2M Client MUST notify the Server with respect to "Period Minimum" rule.								

Note: the following rules MUST be respected ("lt" value + 2*"stp" values < "gt" value)

Table 4: [NOTIFICATION] class Attributes

5.2 Bootstrap Interface

The Bootstrap Interface is used to provision essential information into the LWM2M Client to enable the LWM2M Client to perform the operation "Register" with one or more LWM2M Servers.

There are four bootstrap modes supported by the LWM2M Enabler:

- Factory Bootstrap
- Bootstrap from Smartcard
- Client Initiated Bootstrap
- Server Initiated Bootstrap

The LWM2M Client MUST support at least one bootstrap mode specified in the Bootstrap Interface.

The LWM2M Bootstrap Server MUST support Client Initiated Bootstrap and Server Initiated Bootstrap modes specified in the Bootstrap Interface.

This section describes what information is conveyed across the Bootstrap Interface, where the LWM2M Client puts that information and how to provision the Bootstrap Information for each of these bootstrap modes.

5.2.1 Bootstrap Information

This section specifies the information that needs to be configured in LWM2M Client for connecting to the LWM2M Server(s) or the LWM2M Bootstrap Server. This Bootstrap Information can be available before performing the Bootstrap Sequence described in Section 5.2.3 or obtained as a result of the Bootstrap Sequence.

Bootstrap Information can be categorized into two types:

- LWM2M Server Bootstrap Information
- LWM2M Bootstrap Server Bootstrap Information

The LWM2M Client MUST have the LWM2M Server Bootstrap Information after the Bootstrap Sequence specified in Section 5.2.3.

The LWM2M Client SHOULD have the LWM2M Bootstrap Server Bootstrap Information.

The LWM2M Server Bootstrap Information is used by the LWM2M Client to register and connect to the LWM2M Server

The LWM2M Server Bootstrap Information MUST contain at least a LWM2M Server Account. The LWM2M Server Bootstrap Information MAY additionally contain further Object Instances (e.g., Access Control, Connectivity Object).

The LWM2M Client MAY be configured to use one or more LWM2M Server Account(s).

The LWM2M Client MUST have at most one LWM2M Bootstrap Server Account.

The LWM2M Bootstrap Server Bootstrap Information is used by the LWM2M Client to contact the LWM2M Bootstrap Server in order to get the LWM2M Server Bootstrap Information.

The LWM2M Bootstrap Server Bootstrap Information MUST be a LWM2M Bootstrap Server Account.

Bootstrap Information Type	Entity	Required
The LWM2M Server Bootstrap Information	LWM2M Server Account	Yes*
	Additional Object Instances (e.g., Access Control, Connectivity Object)	No
The LWM2M Bootstrap Server Bootstrap Information	LWM2M Bootstrap Server Account (Security Object instance)	No

Table 5: Bootstrap Information List

*the LWM2M Client MUST have at least one LWM2M Server Account after Bootstrap Sequence specified in 5.2.3

Please note that the LWM2M Client MUST accept Bootstrap Information sent via Bootstrap Interface without processing authorization process specified in Section 7.3.2 Authorization.

5.2.2 Bootstrap Modes

This section of the specification provides description and further information for each of the following Bootstrap Modes:

- Factory Bootstrap
- Bootstrap from Smartcard
- Client Initiated Bootstrap
- Server Initiated Bootstrap

5.2.2.1 Factory Bootstrap

In this mode, the LWM2M Client has been configured with the necessary Bootstrap Information prior to deployment of the device.

5.2.2.2 Bootstrap from Smartcard

When the Device supports a Smartcard, the LWM2M Client MUST retrieve and process the bootstrap data contained in the Smartcard as described in Appendix F. When the bootstrap data retrieval is successful, the LWM2M Client MUST process the bootstrap data from the Smartcard and MUST apply the Bootstrap Information to its configuration.

Due to the sensible nature of the Bootstrap Information, a secure channel SHOULD be established between the Smartcard and the LWM2M Device.

When such a secure channel is established between the Smartcard and the LWM2M Device, this secure channel MUST be based on [GLOBALPLATFORM] procedure, mainly described in Appendix G.

In this mode, the LWM2M Client MUST also ensure that the Bootstrap Information previously retrieved from the Smartcard is unchanged within the Smartcard. If Bootstrap Information is changed, the previous Bootstrap Information MUST be disabled in the LWM2M Client and the LWM2M Client MUST apply the new Bootstrap Information from Smartcard to its configuration.

Disabling the bootstrap data (e.g. removing the Smartcard) within the LWM2M Client requires the Bootstrap Information created from the bootstrap data of the previous Smartcard MUST be deleted.

Checking for Smartcard change and disabling MUST be performed by LWM2M Client, each time a “Register” or “Update” operation take place, with a LWM2M Server provisioned from Smartcard. As usual, the Bootstrap security rules (5.2.4) then apply.

NOTE: Bootstrap Information in Smartcard can be updated by using Smartcard OTA protocol as specified in ETSI TS 102 225 [ETSI TS 102.225] / TS 102 226 [ETSI TS 102 226] and extensions such as 3GPP TS 31.115 [3GPP TS 31.115] / TS 31.116 [3GPP TS 31.116] and 3GPP2 C.S0078-0 [3GPP2 C.S0078-0] / C.S0079-0 [3GPP2 C.S0079-0].

5.2.2.3 Client Initiated Bootstrap

As defined in Section 5.2.3 Bootstrap Sequence, scenarios exist when the LWM2M Server is not configured within the LWM2M Client or attempts to perform the "Register" operation with LWM2M Servers have failed.

When these conditions occur, the Client Initiated Bootstrap mode provides a mechanism for the LWM2M Client to retrieve the Bootstrap Information from a LWM2M Bootstrap Server.

The Client Initiated Bootstrap mode requires having a LWM2M Bootstrap Server Account.

The figure below depicts the Client Initiated Bootstrap flow.

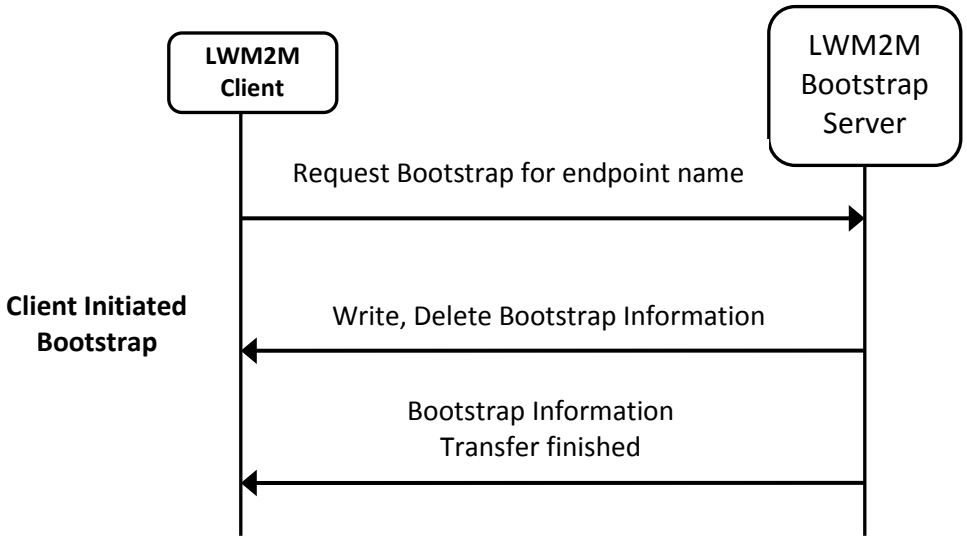


Figure 7: Procedure of Client Initiated Bootstrap

Step #1: Request bootstrap to bootstrap URI

The LWM2M Client sends a "Request Bootstrap" operation to LWM2M Bootstrap Server URI which has been pre-provisioned. When requesting the bootstrap, the LWM2M Client sends the LWM2M Client's "Endpoint Client Name" as a parameter in order to allow the LWM2M Bootstrap Server to provision the proper Bootstrap Information for the LWM2M Client.

Step #2: Configure Bootstrap Information

The LWM2M Bootstrap Server configures the LWM2M Client with the Bootstrap Information using the "Write" and/or "Delete" operation.

The Client Initiated Bootstrap MAY be used to configure some Resources of the Bootstrap Information in the LWM2M Client after initial bootstrap to update Bootstrap Information. In this case, all the Bootstrap Information is OPTIONAL.

Step #3: Bootstrap Finish Indication

When the LWM2M Server has finished to send all the Bootstrap Information to the LWM2M Client, the Server MUST send a Finish Bootstrap Indication to the Client to properly end this phase.

Step #4: Clean-up after successful Bootstrapping

The method of cleaning up the bootstrap credentials after successful bootstrapping depends on whether the LWM2M Bootstrap Server Account contains persistent or non-persistent keys. Persistent keys SHOULD be used for the LWM2M Bootstrap Server Account.

1) LWM2M Bootstrap Server Account contains persistent keys:

   In that case the original LWM2M Bootstrap Server Account SHOULD be kept.

2) LWM2M Bootstrap Server Account contains non-persistent keys:

a) A new LWM2M Bootstrap Server Account MUST be created during the bootstrap process - this time with a persistent keyset - for replacing the original LWM2M Bootstrap Server Account which MUST be purged by the new LWM2M bootstrap server at the end of the bootstrapping, or, only if a) cannot be applied,

b) the LWM2M Client MUST purge the LWM2M Bootstrap Server Account after the timeout value Bootstrap Server Account Timeout, if defined, or immediately after successful bootstrapping if the timeout value is not defined.

Note: Using non-persistent keys for the Bootstrap Server Account is not recommended as explained above. However, in case non-persistent keys are used the following needs to be considered when using the mechanism described in 2b) above:
If the original LWM2M Bootstrap Server Account is purged from the device, and a new LWM2M Bootstrap Server Account is NOT created, further adding or removing of LWM2M Server Accounts will no longer be possible. Furthermore, updating security credentials e.g. X.509 certificates will also no longer be possible.

5.2.2.4 Server Initiated Bootstrap

In this mode, the LWM2M Bootstrap Server configures the Bootstrap Information in the LWM2M Client without the LWM2M Client sending a bootstrap request to the LWM2M Bootstrap Server.

As the LWM2M Client does not initiate the "Request Bootstrap" operation to the LWM2M Bootstrap Server, the LWM2M Bootstrap Server needs to know if a LWM2M Device is ready for bootstrapping before the LWM2M Client can be configured by the LWM2M Bootstrap Server. The mechanism that a LWM2M Bootstrap Server gains this knowledge is implementation specific. A common scenario is that elements in the Network Provider's network informs the LWM2M Bootstrap Server of the LWM2M Device when the LWM2M Device connects to the Network Provider's network.

Once the LWM2M Bootstrap Server has been notified that the LWM2M Device is ready to receive the Bootstrap Information, the LWM2M Bootstrap Server configures the LWM2M Client with the Bootstrap Information using the "Write" and/or "Delete" operation.

The Server Initiated Bootstrap mode requires having the Bootstrap Information for the LWM2M Bootstrap Server. The minimum information that needs to be preloaded for this is the security credentials required for a secure DTLS connection to the LWM2M Bootstrap Server.

The figure below depicts the Server Initiated Bootstrap flow.

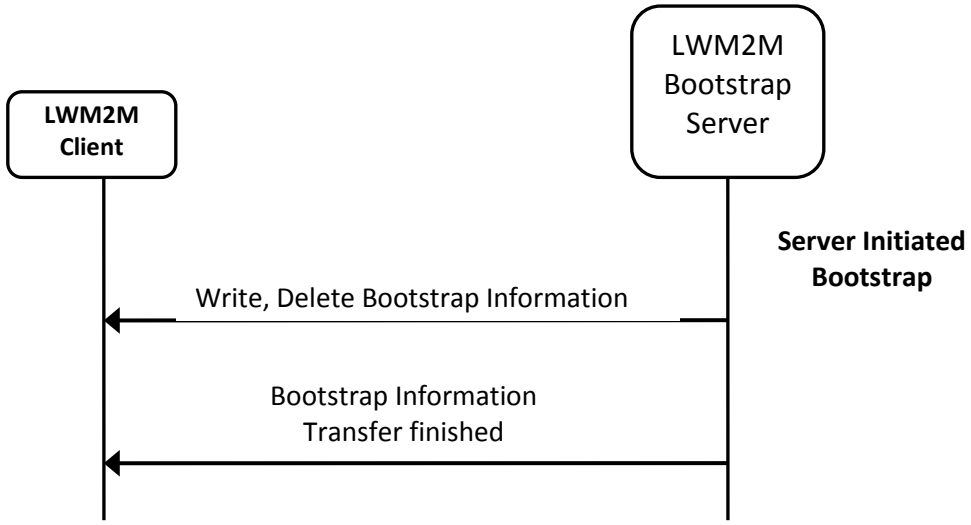


Figure 8: Procedure of Server Initiated Bootstrap

Step #1: Configure Bootstrap Information

The LWM2M Bootstrap Server configures the Bootstrap Information in the LWM2M Client using the "Write" and/or "Delete" operation.

The Server Initiated Bootstrap MAY be used to configure some Resources of the Bootstrap Information in the LWM2M Client after initial bootstrap to update Bootstrap Information. In this instance, all the Bootstrap Information are OPTIONAL.

Step #2: Bootstrap Finish Indication

When the LWM2M Server has finished to send all the Bootstrap Information to the LWM2M Client, the Server MUST send a Finish Bootstrap Indication to the Client to properly end this phase.

Step #3: Clean-up after successful Bootstrapping

The method of cleaning up the bootstrap credentials after successful bootstrapping depends on whether the LWM2M Bootstrap Server Account contains persistent or non-persistent keys. Persistent keys SHOULD be used for the LWM2M Bootstrap Server Account.

1) LWM2M Bootstrap Server Account contains persistent keys:

   In that case the original LWM2M Bootstrap Server Account SHOULD be kept.

2) LWM2M Bootstrap Server Account contains non-persistent keys:

   a) A new LWM2M Bootstrap Server Account MUST be created during the bootstrap process - this time with a persistent keyset - for replacing the original LWM2M Bootstrap Server Account which MUST be purged by the new LWM2M bootstrap server at the end of the bootstrapping, or, only if a) cannot be applied,

   b) the LWM2M Client MUST purge the LWM2M Bootstrap Server Account after the timeout value Bootstrap Server Account Timeout, if defined, or immediately after successful bootstrapping if the timeout value is not defined.

Note: Using non-persistent keys for the Bootstrap Server Account is not recommended as explained above. However, in case non-persistent keys are used the following needs to be considered when using the mechanism described in 2b) above:
If the original LWM2M Bootstrap Server Account is purged from the device, and a new LWM2M Bootstrap Server Account is NOT created, further adding or removing of LWM2M Server Accounts will no longer be possible. Furthermore, updating security credentials e.g. X.509 certificates will also no longer be possible.

5.2.3 Bootstrap Sequence

The LWM2M Client MUST respect step by step the procedural sequence specified below when attempting to

bootstrap a LWM2M Device:

1. If the LWM2M Device has Smartcard, the LWM2M Client tries to obtain Bootstrap Information from the Smartcard using the Bootstrap from Smartcard mode.
   Any Server Initiated Bootstrap attempt MUST be ignored by the LWM2M Client until it has tried to bootstrap via Smartcard or Factory Bootstrap mode.
2. If the LWM2M Client is not configured using the Bootstrap from Smartcard mode, the LWM2M Client tries to obtain the Bootstrap Information by using Factory Bootstrap mode.
   Any Server Initiated Bootstrap attempt MUST be ignored by the LWM2M Client until it has tried to bootstrap via Smartcard or Factory Bootstrap mode.
3. If the LWM2M Client has any LWM2M Server Object Instances from the previous steps, the LWM2M Client tries to register to the LWM2M Server(s) configured in the LWM2M Server Object Instance(s).
4. If LWM2M Client fails to register to all the LWM2M Servers or the Client doesn't have any LWM2M Server Object Instances, and the LWM2M Client hasn't received a Server Initiated Bootstrap within the ClientHoldOffTime, the LWM2M Client performs the Client Initiated Bootstrap.
5. A Server Initiated Bootstrap attempt (e.g. for updating an LWM2M Server Account) remains possible, but only if the LWM2M Client retains the corresponding LWM2M Bootstrap Server Account.

5.2.4 Bootstrap Security

The information conveyed through the Bootstrap Interface is sensitive and requires that communication session, security mechanisms and/or keys MUST be different instances from the one that is used for the other LWM2M Interfaces.

If the LWM2M Client or the LWM2M Bootstrap Server needs to convey Bootstrap Information across the Bootstrap Interface, the LWM2M Client or the LWM2M Bootstrap Server MUST establish a new secure communication session.

If security materials (e.g. LWM2M Server URI, Security Mode, and Security Key), are changed in the LWM2M Client, the LWM2M Client MUST disconnect the existing communication session between the LWM2M Server and LWM2M Client and establish a new secure communication session between the LWM2M Server and LWM2M Client using the security mechanism and/or keys which have been configured by Bootstrap Interface.

5.2.5 Bootstrap Commands

The mapping to CoAP Methods of the LWM2M Bootstrap Interface operations specified in this section, is detailed in chapter 8 of the present document (Transport Layer Binding and Encodings).

5.2.5.1 BOOSTRAP WRITE

The “Write” operation in Bootstrap Interface is different from the “Write” Operation in Device Management and Service Enablement interface; the LWM2M Client MUST write the value included in the payload regardless of an existence of the targeting Object Instance(s) or Resource(s).

The Write operation can be sent multiple times.

Only in Bootstrap Interface, the “Write” MAY target just an Object ID, which will allow a Bootstrap Server in using a TLV or JSON formatted payload, to populate a LWM2M Client in a single message containing several Instances of the same Object.

The Bootstrap “Write” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance to write. If no Object Instance ID is indicated , Object Instance(s) MUST be specified in the TLV or JSON payload
Resource ID	No	-	Indicates the Resource to write. The payload is the new value for the Resource. If no Resource ID is indicated, then the value included payload is an Object Instance containing the Resource values.
New Value	Yes	-	The new value included in the payload to update the Object Instance(s) or Resource.

5.2.5.2 BOOTSTRAP DELETE

The Delete operation targets an Object Instance and can be sent multiple times.

Only in Bootstrap Interface, the Delete operation MAY targets “/” URI. In that case, the operation MUST delete all the existing Object Instances - except LWM2M Bootstrap Server Account - in the LWM2M Client; this functionality could be used for initialization purpose before LWM2M Bootstrap Server sends Write operation(s) to the LWM2M Client.

The Delete operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	No	-	Indicates the Object from which Object Instance will be deleted; if no Object ID is indicated, all existing Object Instances (except the LWM2M Bootstrap Server Account one) in the LWM2M Client will be deleted
Object Instance ID	No	-	Indicates the Object Instance to delete (Object ID MUST be provided)

5.2.5.3 BOOTSTRAP REQUEST

The Bootstrap Request operation is only performed to initiate the Bootstrap Sequence in the "Client Initiated Bootstrap" mode.

The Bootstrap Request operation has the following parameter:

Parameter	Required	Default Value	Notes
/bs?ep={Endpoint Client Name}	Yes	-	Indicates the LWM2M Client's "Endpoint Name" in order to allow the LWM2M Bootstrap to provision the Bootstrap Information for the LWM2M Client

5.2.5.4 BOOTSTRAP FINISH

The Bootstrap Finish operation is only performed to terminate the Bootstrap Sequence previously initiated either in "Client Initiated Bootstrap" mode or in "Server Initiated Bootstrap" mode.

This command informs the LWM2M Client, that all the Bootstrap Information have been provided by the LWM2M Bootstrap Server

The Bootstrap Finish operation has the following parameter:

Parameter	Required	Default Value	Notes
/bs	Yes	-	-

5.3 Client Registration Interface

The LWM2M Server MUST support all the operations in this interface and the LWM2M Client MUST support "Register" and "Update" and SHOULD support "De-register" operation.

The Client Registration Interface[1] is used by a LWM2M Client to register with one or more LWM2M Servers, maintain each registration and de-register from a LWM2M Server. The registration is based on the Resource Model and Identifiers defined in Section 6 Identifiers and Resources. When registering, the LWM2M Client performs the "Register" operation and provides the properties the LWM2M Server requires to contact the LWM2M Client (e.g., End Point Name); maintain the registration and session (e.g., Lifetime, Queue Mode) between the LWM2M Client and LWM2M Server as well as knowledge of the Objects the LWM2M Client supports and existing Object Instances in the LWM2M Client. The registration is soft state, with a lifetime indicated by the Lifetime Resource of that LWM2M Server Object Instance. The LWM2M Client periodically performs an update of its registration information to the registered LWM2M Server(s) by performing the "Update" operation. If the lifetime of a registration expires without receiving an update from the LWM2M Client, the LWM2M Server removes the registration. Finally, when shutting down or discontinuing use of a LWM2M Server, the LWM2M Client performs a "De-register" operation.

The Binding Resource of the LWM2M Server Object informs the LWM2M Client of the transport protocol preferences of the LWM2M Server for the communication session between the LWM2M Client and LWM2M Server. The LWM2M Client SHOULD perform the operations with the modes indicated by the Binding Resource of the LWM2M Server Object Instance.

[1] The mapping to CoAP Methods of the LWM2M Client Registration Interface operations specified in this section, is detailed in chapter 8 of the present document (Transport Layer Binding and Encodings).

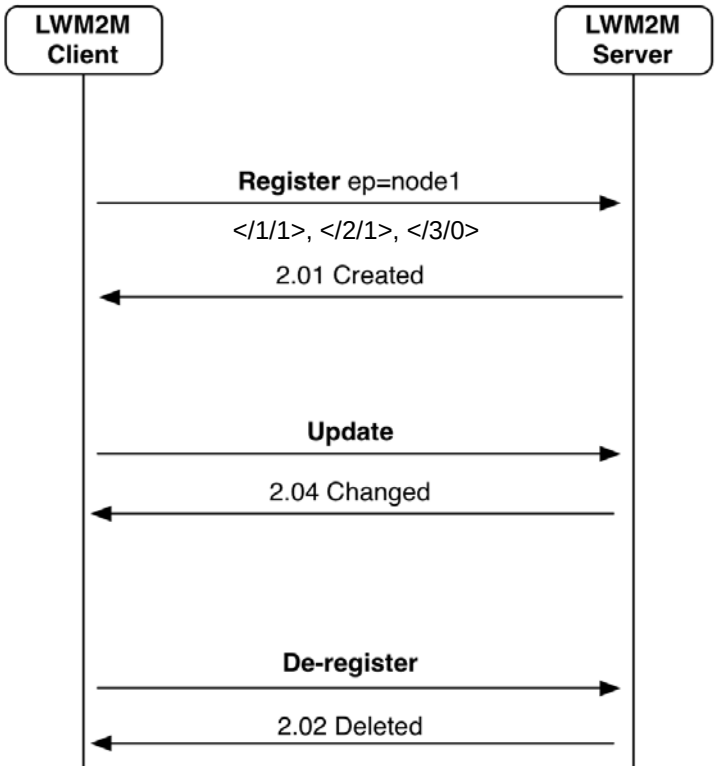


Figure 9: Client Registration Interface example flows.

5.3.1 Register

Registration is performed when a LWM2M Client sends a "Register" operation to the LWM2M Server. After the LWM2M Device is turned on and the bootstrap procedure has been completed, the LWM2M Client MUST perform a "Register" operation to each LWM2M Server that the LWM2M Client has a Server Object Instance. Table 6 describes the parameters used for the "Register" operation.

The "Register" operation includes the Endpoint Client Name parameter along with other parameters listed in Table 6. The "Register" operation MUST include a value for the Endpoint Client Name parameter that is unique on that LWM2M Server.

Upon receiving a "Register" operation from the LWM2M Client, the LWM2M Server records the IP address and port from the IP packet of the registration message and uses this information for all future interactions with that LWM2M Client.

If the LWM2M Client sends a "Register" operation to the LWM2M Server even though the LWM2M Server has registration information of the LWM2M Client, the LWM2M Server removes the existing registration information and performs the new "Register" operation. This situation happens when the LWM2M Client forgets the state of the LWM2M Server (e.g., factory reset).

The LWM2M Server MUST support all the parameters listed at Table 6 and the LWM2M Client MUST support Endpoint Client Name, Lifetime, Binding Mode, and Object and Object Instances and MAY support LWM2M Version and SMS Number.

Parameter	Required	Default Value	Notes
Endpoint Client Name	Yes		See Section 6.2
Lifetime	No	86400	If Lifetime Resource does not exist in a LWM2M Server Object Instance (see Appendix D.1), the Client MUST NOT send this parameter and the Server MUST regard lifetime of the Client as 86400 seconds The registration SHOULD be removed by the Server if a new registration or update is not received within this lifetime.

Parameter	Required	Default Value	Notes
LWM2M Version	No	1.0	Indicates the version of the LWM2M Enabler that the LWM2M Client supports. This parameter is required only for LWM2M versions > 1.0.
Binding Mode	No	U	Indicates current binding and Queue mode of the LWM2M Client. “U” means UDP binding, and "S" means SMS binding. The "Q" can be appended to represent the binding works in the Queue mode. For example, "UQS" means the Client uses both the UDP binding with Queue Mode enabled and the SMS binding with Queue Mode disabled. The valid values of the parameter are listed in the Section 5.3.1.1.
SMS Number	No		The value of this parameter is the MSISDN where the LWM2M Client can be reached for use with the SMS binding.
Objects and Object Instances	Yes		The list of Objects supported and Object Instances available on the LWM2M Client.

Table 6: Registration parameters

The list of Objects and Object Instances is included in the payload of the registration message. The payload Media-Type of that registration message MUST be the Core Link Format (application/link-format) defined in [RFC6690], so that each Object is described as a Link according to that format. The Target component of the link is required, and consists of the Object path. Any other parameters included in the link MUST be silently ignored, unless specified for use by the LWM2M Enabler.

The payload for a LWM2M Client supporting LWM2M Server, Access Control, Device, Connectivity Monitoring and Firmware Update Objects from Appendix E would simply be:

</1>, </2>, </3>, </4>, </5>

If Objects Instances are already available on the LWM2M Client at the time of registration, then the format would be (for the example client of Appendix F):

</1/0>, </1/1>, </2/0>, </2/1>, </2/2>, </3/0>, </4/0>, </5>

By default, the RFC6690 links of Objects are located under the root path as in the example above. However, devices might be hosting other Resources on an endpoint, and there may be the need to place Objects under an alternative path. This is achieved by including an OMA LWM2M link in addition to the Object links as follows, e.g. to place Objects under the “/lwm2m” path:

</lwm2m>;rt="oma.lwm2m", < /1/101>, < /1/102>, < /2/0>, < /2/1>, < /2/2>, < /3/0>,< /4/0>,< /5>

The RFC6690 Resource Type parameter (i.e., rt="oma.lwm2m") MAY be used to provide the information that the path in front of the Resource Type parameter is used for the LWM2M enabler.

The Resource Type value “oma.lwm2m” has to be registered with the appropriate IANA registry for this purpose.

If the LWM2M Client supports the JSON data format for all the Objects it SHOULD inform the LWM2M Server by including the content type in the root path link using the ct= link attribute. An example is as follows (note that the content type value 1543 is an example, the actual value will be assigned by IANA for the LWM2M JSON format).

</>;ct=1543, </1/0>, </1/1>, </2/0>, </2/1>, </2/2>, </3/0>, </4/0>, </5>

5.3.1.1 Behavior with Current Transport Binding and Mode

Behavior of the LWM2M Server and the LWM2M Client is differentiated by Current Transport Binding and Mode. Current Transport Binding and Mode is decided by “Binding” Resource set by the LWM2M Server and whether SMS and/or Queue Mode are supported by the LWM2M Client. Queue Mode is useful when the LWM2M Device is not reachable by the LWM2M Server at all the times and it could help the LWM2M Client sleep longer. Table 7 describes the behaviour of the LWM2M Server and the LWM2M Client for each Current Transport Binding and Mode.

Current Transport Binding and Mode	Behaviour
U (UDP)	The LWM2M Server expects that the LWM2M Client is reachable via the UDP binding at any time. The LWM2M Server MUST send requests to a LWM2M Client using the UDP binding. The LWM2M Client MUST send the response to such a request over the UDP binding. This is the normal default mode of operation.
UQ (UDP with Queue Mode)	The Server MUST queue all requests to the LWM2M Client, sending requests via UDP when the LWM2M Client is on-line as described in Section 8.4 Queue Mode Operation. The LWM2M Server MUST send requests to a LWM2M Client using the UDP binding. The LWM2M Client MUST send the response to such a request over the UDP binding.
S (SMS)	The LWM2M Server expects that the LWM2M Client is reachable via the SMS binding at any time. The LWM2M Server MUST send requests to a LWM2M Client using the SMS binding. The LWM2M Client MUST send the response to such a request over the SMS binding.
SQ (SMS with Queue Mode)	The Server MUST queue all requests to the LWM2M Client, sending requests via SMS when the LWM2M Client is on-line as described in Section 8.4 Queue Mode Operation. Requests MUST be sent to the LWM2M Client using the SMS binding. The LWM2M Client MUST send the response to such a request over the SMS binding.
US (UDP and SMS)	The LWM2M Server expects that the LWM2M Client is reachable via the UDP binding at any time. The LWM2M Server expects that the LWM2M Client is reachable via the SMS binding at any time. If the LWM2M Server sends requests to a LWM2M Client using the UDP binding, The LWM2M Client MUST send the response to such a request over the UDP binding. If the LWM2M Server sends requests to a LWM2M Client using the SMS binding, The LWM2M Client MUST send the immediate response to such a request over the SMS binding.
UQS (UDP with Queue Mode and SMS)	The Server MUST queue all requests to the LWM2M Client, sending requests via UDP when the LWM2M Client is on-line as described in Section 8.4 Queue Mode Operation. The LWM2M Server expects that the LWM2M Client is reachable via the SMS binding at any time. If the LWM2M Server sends requests to a LWM2M Client using the UDP binding, The LWM2M Client MUST send the response to such a request over the UDP binding. If the LWM2M Server sends requests to a LWM2M Client using the SMS binding, The LWM2M Client MUST send the immediate response to such a request over the SMS binding. The LWM2M Server MAY request the LWM2M Client to perform “Update” operation via UDP by sending “Execute” operation on “Registration Update Trigger” Resource via SMS.

Table 7: Behaviour with Current Transport Binding and Mode

UQSQ and USQ are not supported.

5.3.2 Update

Periodically or based on certain events within the LWM2M Client or initiated by the LWM2M Server, the LWM2M Client updates its registration information with a LWM2M Server by sending an "Update" operation to the LWM2M Server. This "Update" operation MUST contain only the parameters listed in Table 8 which have changed compared to the last registration parameters sent to the LWM2M Server.

If the LWM2M Client is using the UDP binding to communicate with a LWM2M Server and LWM2M Client's IP address or the port changes, the LWM2M Client MUST send an "Update" operation to the LWM2M Server.

Parameter	Required
Lifetime	No
Binding Mode	No
SMS Number	No
Objects and Object Instances	No

Table 8: Update parameters

The "Update" operation can be initiated by the LWM2M Server via an "Execute" operation on the "Registration Update Trigger" Resource of the LWM2M Server Object.

5.3.3 De-register

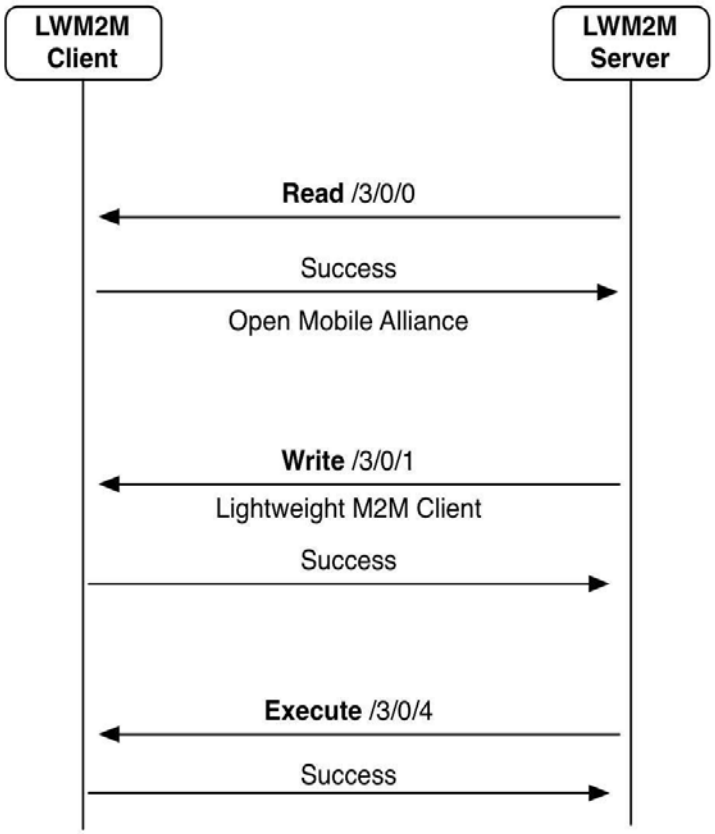
When a LWM2M Client determines that it no longer requires to be available to a LWM2M Server (e.g., LWM2M Device factory reset), the LWM2M Client SHOULD send a "De-register" operation to the LWM2M Server. Upon receiving this message, the LWM2M Server removes the registration information from the LWM2M Server.

5.4 Device Management & Service Enablement Interface

The LWM2M Server and the LWM2M Client MUST support all the operations in this interface.

The Device Management and Service Enable Interface[2] is used by the LWM2M Server to access Object Instances and Resources available from the LWM2M Client. The interface provides this access through the use of "Create", "Read", "Write", "Delete", "Execute", "Write Attributes", or "Discover" operations. The operations that Resource supports are defined in the Object definition using the Object Template. The Object Template is described in Appendix D.1 Object Template. The Normative Objects defined by the LWM2M Enabler are described in Appendix E.

[2] The mapping to CoAP Methods of the LWM2M Client Registration Interface operations specified in this section, is detailed in chapter 8 of the present document (Transport Layer Binding and Encodings).



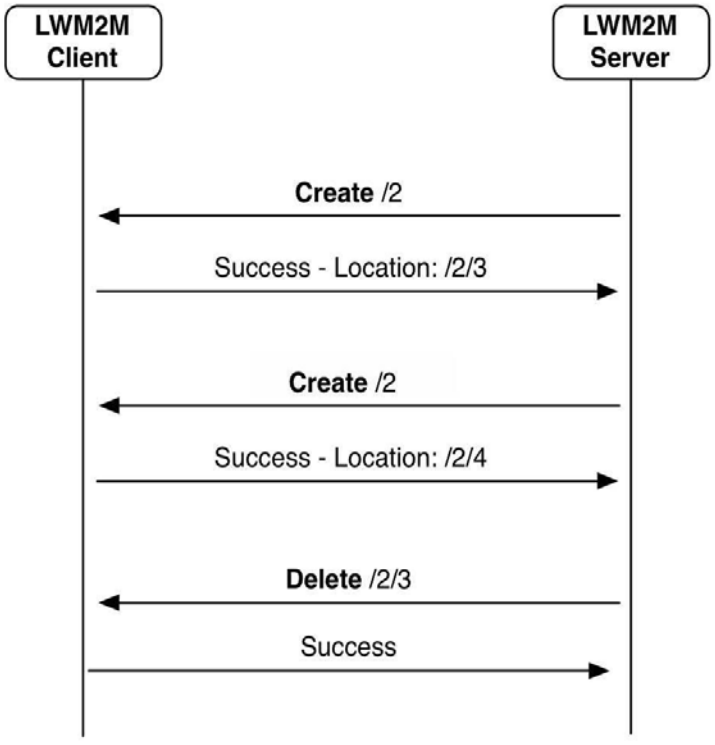


Figure 10: Example flows of Device Management & Service Enablement Interface

5.4.1 Read

The “Read” operation is used to access the value of a Resource, an array of Resource Instances, an Object Instance or all the Object Instances of an Object. The “Read” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance to read. If no Object Instance ID is indicated, then the Object Instances of the Object, which the Server is authorized to, are returned.
Resource ID	No	-	Indicates the Resource to read. If no Resource ID is indicated, then the whole Object Instance is returned.

Table 9: Read parameters

5.4.2 Discover

The “Discover” operation is used to discover LWM2M Attributes attached to an Object, Object Instances, and Resources. This operation can be used to discover which Resources are implemented for a given Object Instance. The returned payload is a list of application/link-format CoRE Links [RFC6690] for each targeted Object, Object Instance, or Resource, along with their attached Attributes.

The “Discover” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance.
Resource ID	No	-	Indicates the Resource.

Table 10: Discover parameters

If Object ID is only specified, the LWM2M Client MUST respond to the “Discover” operation with the list of Object Instances and the list of their respective Resources implemented by the LWM2M Client for that Object. In addition the list of Attributes which have be assigned to this Object level (see section 5.3.1.1) are also returned.
For example:

- when the “Discover” operation targets an Object with Object ID of 3, the response to the operation could be:

  </3>;pmin=10, </3/0/1>, <3/0/2>, </3/0/3>, </3/0/4>, <3/0/6>,<3/0/7>,<3/0/8>,<3/0/11>,<3/0/16>which means that the LWM2M Client implements the Device Info Object (Instance 0)with Resource IDs of 1,2,3, 4 6,7,8,11,and 16 among the Resources of Devive Info Object, with anR-Attributes assigned to the Object level.

- when the “Discover”operation targets an Object ID and Object Instance ID only, the list of Attributes assigned to that Object Instance MUST be reported, and the list of implemented Resources and their attached Attribute MUST be returned in the response as well. For example: if Object ID is 3 and Object Instance ID is 0, then

  </3/0>;pmax=60, </3/0/1>, <3/0/2>, </3/0/3>, </3/0/4>, <3/0/6>;dim=8,

  <3/0/7>;dim=8; gt=50;lt=42.2,<3/0/8>;dim=8,<3/0/11>,<3/0/16>

  means that regarding the Device Info Object Instance, an R-Attribute has been assigned to this Instance level. And the LWM2M Client implements the multiple Resources 6,7, and 8 with a dimension of 8 and supports 2 additional Notification parameters for Ressource 7

- when the “Discover”operation targets an Object ID, Object Instance ID and Resource ID, the attributes of that Resource MUST be returned. In addition, the R-Attributes inherited from upper levels (Object and Object Instance level) are also reported for that Resource (The rules of Section 5.3.1.1 apply) For example: if Object ID is 3, Object Instance ID is 2, and Resource ID is 7, then

</3/2/7>;dim=8;pmin=10;pmax=60;gt=50;lt=42.2 with pmin assigned at the Object level, and pmax assigned at the Object Instance level

If a Resource, an Object Instance, or an Object has attributes for multiple LWM2M Servers, then one link is returned for each and the ep= attribute is used to indicate the Short Server ID of the LWM2M Server. For example: if Object ID is 3 and Object Instance ID is 0, and Resource ID is 7 with two Observe operations from two Servers, then

</3/0/7>;ep=1;dim=8;gt=50;lt=42.2,
</3/0/7>;ep=2;dim=8;pmax=300;gt=80;lt=75.5

5.4.3 Write

The "Write" operation is used to change the value of a Resource, the values of an array of Resources Instances or the values of multiple Resources from an Object Instance.

The request includes the value to be written in the corresponding Plain Text, Opaque, TLV or JSON format according to the Content-Format option which MUST be specified [CoAP].

When more than a single value of a Resource has to be changed in a "Write" request, the TLV or JSON Content Format MUST be used.

The Write request MUST be rejected when the specified Content Format is not supported by the LWM2M Client.

LWM2M Client and LWM2M Server MUST support the following mechanisms to change multiple Resources or an array of Resource Instances:

- Replace: replaces the Object Instance or the Resource(s) with the new value provided in the "Write" operation.
- Partial Update: adds or updates Resources provided in the new value and leaves other existing Resources unchanged.

The "Write" operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	Yes	-	Indicates the Object Instance to write.
Resource ID	No	-	Indicates the Resource to write. The payload is the new value for the Resource. If no Resource ID is indicated, then the value included payload is an Object Instance containing the Resource values.
New Value	Yes	-	The new value included in the payload to update the Object Instance or a single Resource.

Table 11: Write parameters

5.4.4 Write Attributes

In LWM2M 1.0, only Attributes from the [NOTIFICATION] class MAY be changed in using the "Write Attributes" operation.

The general rules for Attributes which are specified in Section 5.1.1 fully apply here. Table 3 in Section 5.1.2 provides explanation on the Attributes supported by the "Write Attributes" operation: Minimum Period, Maximum Period, Greater Than, Less Than, and Step.

The operation permits multiple attributes to be modified within the same operation.

Including [NOTIFICATION] class Attributes specified in Table 3 Section 5.3.1, the "Write Attribute" operation has the following parameters:

Note: How to indicate the Attributes in the message payload is specified in [CoRE_Interface].

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance.
Resource ID	No	-	Indicates the Resource.
[NOTIFICATION] class Attributes	No		

Table 12: Write Attributes parameters

5.4.5 Execute

The “Execute” operation is used by the LWM2M Server to initiate some action, and can only be performed on individual Resources. A LWM2M Client MUST return an error when the “Execute” operation is received for an Object Instance(s) or Resource Instance(s).

Resource which supports “Execute” operation MAY have arguments.

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	Yes	-	Indicates the Object Instance.
Resource ID	Yes	-	Indicates the Resource to execute.
Arguments	No	-	The arguments of the “Execute” operation are expressed in Plain Text format (syntax bellow)

Table 13: Execute parameters

In using ABNF, the syntax of the arguments, and arguments list is given as follows:

```
arglist = arg *( "," arg )

arg = DIGIT / DIGIT "=" "'" *CHAR "'"
DIGIT = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
CHAR = "!" / %x23-26 / %x28-5B / %x5D-7E
```

Examples of valid lists of arguments

a) 5

b) 2=’10.3’

c) 7, 0=’ https://www.oma.org’,

d) 0,1,2,3,4

5.4.6 Create

The “Create” operation is used by the LWM2M Server to create Object Instance(s) within the LWM2M Client. The “Create” operation MUST target an Object, and MUST follow the rules specified in section 7.3 (ACCESS CONTROL) and its sub-sections. If any error occurs, nothing MUST be created.

The Object Instance created in the LWM2M Client by the LWM2M Server MUST be an Object type supported by the LWM2M Client and announced to the LWM2M Server using the “Register” and “Update” operations of the LWM2M Client Registration Interface.

Object Instance whose Object supports at most one Object Instance MUST be assigned an Object Instance ID of 0 when the Object Instance is Created.

The “Create” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
New Value	Yes	-	The new value included in the payload to create the Object Instance(s)

Table 14: Create parameters

The new value included in the payload MUST follow the TLV or JSON format according to the Content-Format option which MUST be specified.

When there is no reference to Object Instance in the TLV/JSON payload of the “Create” command, the LWM2M Client MUST assigns the ID of the created Object Instance. If a new Object Instance is created through that operation and the LWM2M Client has more than one LWM2M Server Account, then the LWM2M Client creates an Access Control Object Instance for the created Object Instance (7.3 ACCESS CONTROL)

- Access Control Owner MUST be the LWM2M Server
- The LWM2M Server MUST have full access rights

5.4.7 Delete

The “Delete” operation is used for LWM2M Server to delete an Object Instance within the LWM2M Client.

The Object Instance that is deleted in the LWM2M Client by the LWM2M Server MUST be an Object Instance that is announced by the LWM2M Client to the LWM2M Server using the “Register” and “Update” operations of the Client Registration Interface.

The Delete operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	Yes	-	Indicates the Object Instance to delete.

Table 15: Delete parameters

5.5 Information Reporting Interface

The LWM2M Server and the LWM2M Client MUST support all the operations in this interface.

The Information Reporting Interface[3] is used by a LWM2M Server to observe any changes in a Resource on a LWM2M Client, receiving notifications when new values are available. This observation relationship is initiated by sending an “Observe” operation to the L2M2M Client for an Object, an Object Instance or a Resource. An observation ends when a “Cancel Observation” operation is performed.

[3] The mapping to CoAP Methods of the LWM2M Information Reporting Interface operations specified in this section, is detailed in chapter 8 of the present document (Transport Layer Binding and Encodings).

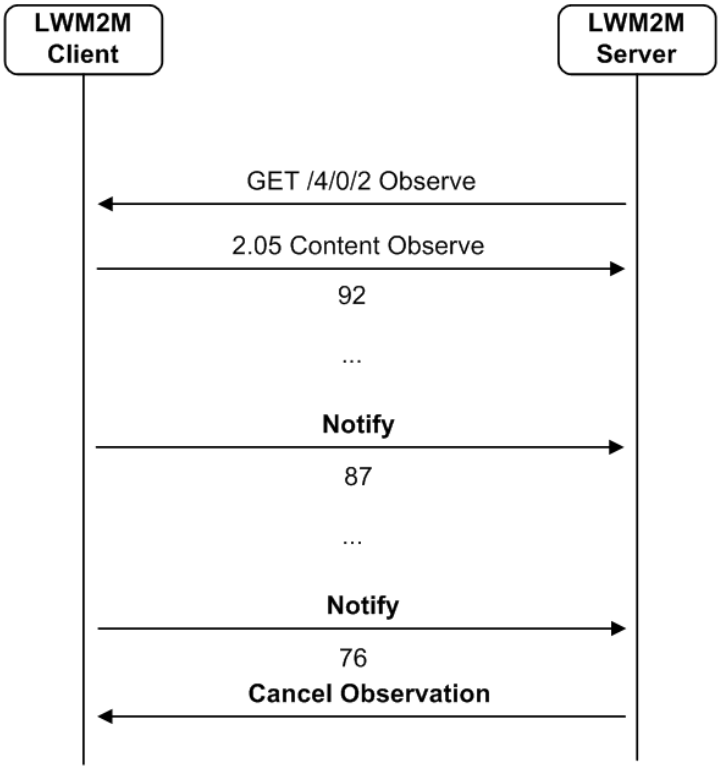


Figure 11: Example flow for Information Reporting Interface for the RSSI Resource of the Connectivity Monitoring Object of the example client (Appendix E).

5.5.1 Observe

The LWM2M Server initiates an observation request for changes of a specific Resource, Resources within an Object Instance or for all the Object Instances of an Object within the LWM2M Client.

Related parameters for "Observe" operation are described in 5.4.4 Write Attributes and those parameters are configured by "Write Attributes" operation.

The Observe operation includes the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance to observe. If no Object Instance ID is indicated, then all the Object Instances of Objects are observed and Resource ID MUST NOT be specified.
Resource ID	No	-	Indicates the Resource to observe. If no Resource ID is indicated, then the whole Object Instance is observed.

Table 16: Observe parameters

When "Observe" operation contains only Object ID, the "Notify" operation MUST be done per Object Instance.

5.5.2 Notify

The "Notify" operation is sent from the LWM2M Client to the LWM2M Server during a valid observation on an Object Instance or Resource. This operation includes the new value of the Object Instance or Resource. The "Notify" operation SHOULD be sent when all the conditions (i.e., Minimum Period, Maximum Period, Greater Than, Less Than, Step) configured by "Write Attributes" operation for "Observe" operation are met.

Parameter	Required	Default Value	Notes
Updated Value	Yes	-	The new value included in the payload about the Object Instance or Resource.

Table 17: Notify parameters

The following example shows how the Minimum and Maximum period parameters work as shown in Section 5.4.4. A LWM2M Server makes an observation for a Temperature Resource that is updated inside the LWM2M Client at irregular periods (based on change). The LWM2M Server makes an observation when the Minimum Period = 10 Seconds and Maximum Period = 60 Seconds have been set for that Resource. The LWM2M Client will wait at least 10 Seconds before sending a "Notify" operation to the LWM2M Server (even if the Resource has changed before that), and no longer than 60 Seconds before sending a "Notify" operation (even if the Resource has not changed yet). The "Notify" operation is sent anywhere between 10-60 seconds upon change.

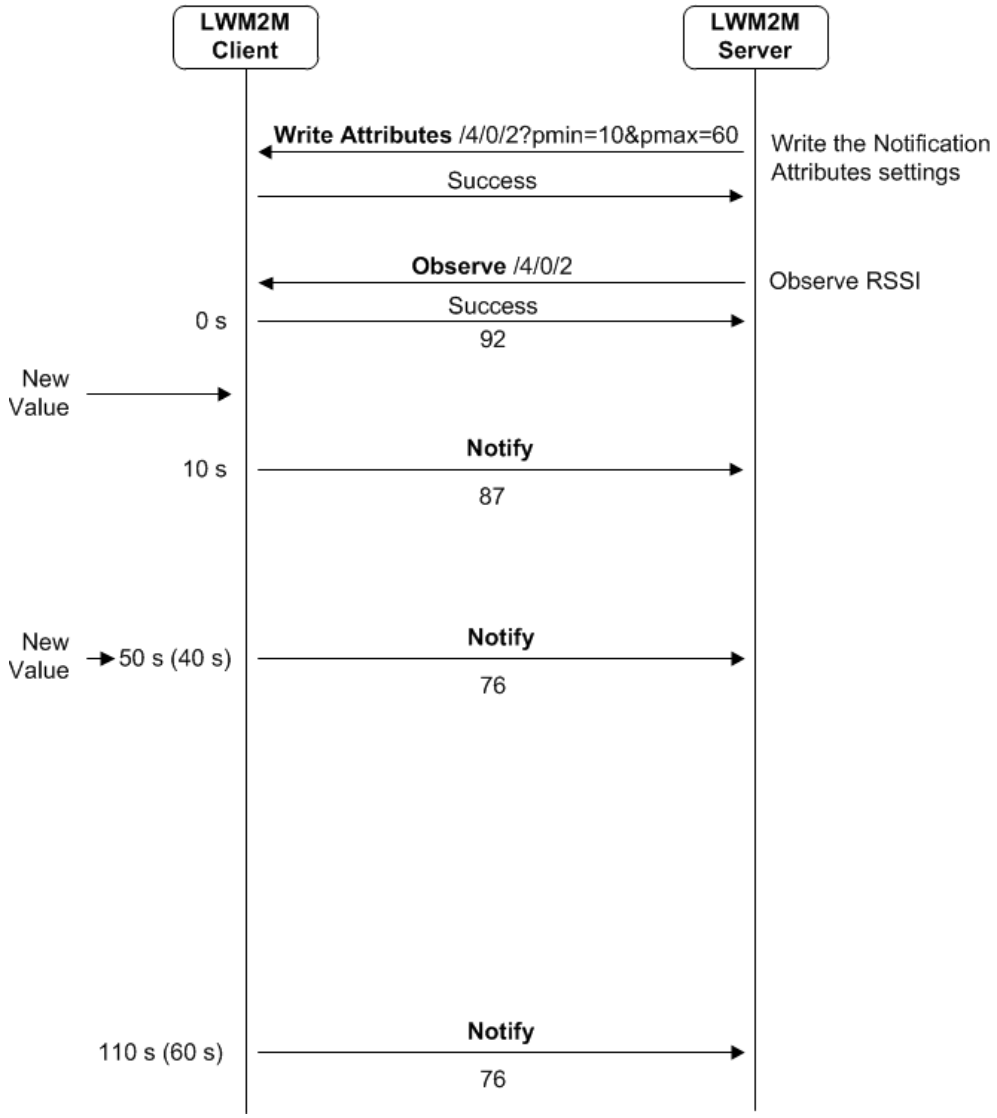


Figure 12: Example of Minimum and Maximum periods in an Observation.

This example assumes the Minimum Period has been set to 10 and the Maximum Period set to 60 for the Resource /4/0/2 before making the observation.

5.5.3 Cancel Observation

The “Cancel Observation” operation is sent from the LWM2M Server to the LWM2M Client to end an observation relationship for Object Instance or Resource.

Please note that this enabler provides two ways for the LWM2M Server to cancel observation:

1. in response to a “Notify” operation for which it is not interested in any more, the LWM2M Server can send a “Reset Message”.
2. at any moment, by sending a GET request with Observe option=1, the LWM2M Server can cancel an “Observe” operation on a specified Resource, or specified Object Instance(s) .

6. Identifiers and Resources

This section defines the identifiers and resource model for the LWM2M Enabler.

6.1 Resource Model

The LWM2M Enabler defines a simple resource model where each piece of information made available by the LWM2M Client is a Resource. Resources are logically organized into Objects. Figure 13 illustrates this structure, and the relationship between Resources, Objects, and the LWM2M Client. The LWM2M Client may have any number of Resources, each of which belongs to an Object. Resources and Objects have the capability to have multiple instances of the Resource or Object.

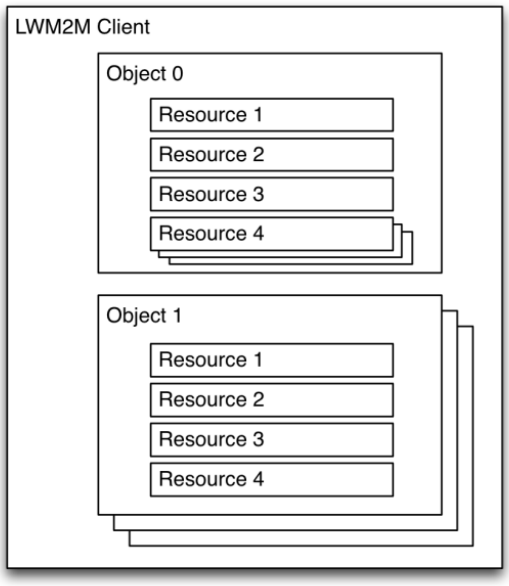


Figure 13: Relationship between LWM2M Client, Object, and Resources

Resources are defined per Object, and each Resource is given a unique identifier within that Object. Each Object and Resource is defined to have one or more operations that it supports. A Resource MAY consist of multiple instances called a Resource Instance as defined in the Object specification. The LWM2M Server can send “Write” operation with JSON or TLV format to Resource to instantiate a Resource Instance. The LWM2M Client also has the capability to instantiate a Resource Instance.

An Object defines a grouping of Resources, for example the Firmware Update Object contains all the Resources used for firmware update purposes. Each Object is assigned a unique OMA Management Object identifier and corresponding index which identifies an Object defined for the LWM2M Enabler. The LWM2M Enabler defines standard Objects and Resources. Further Objects may be added by OMA or other organizations to enable additional M2M Services.

An Object MUST be instantiated either by the LWM2M Server or the LWM2M Client, which is called Object Instance before using the functionality of an Object. After an Object Instance is created, the LWM2M Server can access that Object Instance and Resources which belong to that Object Instance.

The LWM2M Server performs operations on an Object, Object Instance and Resources as described in Section 5 Interfaces. These operations are conveyed as described in Section 7 Transport Layer Binding and Encoding and how to convey the Operation data is defined in 6.3.

The LWM2M Enabler defines an access control mechanism per Object Instance. Object Instances SHOULD have an associated Access Control Object Instance. An Access Control Object Instances contains Access Control Lists (ACLs) that define which operations on a given Oject Instance are allowed for which LWM2M Server(s).

Figure 14 shows an example of the operations the Resources support and how Object Instances and Resources are associated with Access Control Object Instance. In the example, Object Instance 0 for Object 0 has 2 Resources. Resource 1 supports the ”Read”, “Write” and ”Execute” operations, while Resource 2 supports only the “Read” operation. The associated Access

Control Object Instance has ACL of Object Instance 0 for Object 0. Server1 is authorized to perform “Read” and “Write” operations to the Object Instance 0 for Object 0 and Resources of the Object Instance. However, due to the supported operations of each Resource, Server1 can perform the “Read” operation on Resource 1 and 2, and also can perform the “Write” and “Execute” operations on Resource 1, but Server1 cannot perform the “Write” operation on Resource 2 and cannot perform the “Execute” operation on both Resources. The detail access control mechanism is defined in Section 7.3 Access Control.

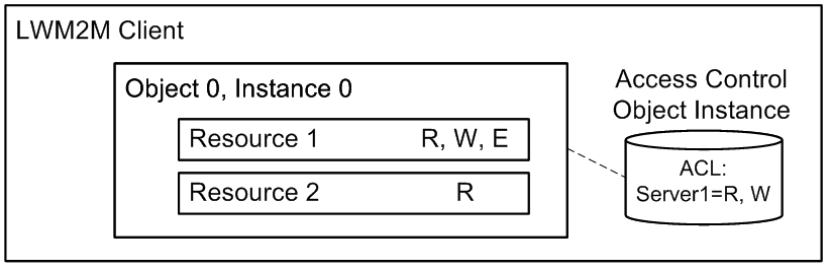


Figure 14: Example of Supported operations and Associated Access Control Object Instance

6.2 Identifiers

The LWM2M Enabler defines specific identifiers for entities used within the LWM2M Protocol. These identifiers are defined in Table 18.

Identifier	Semantics	Description
Endpoint Client Name	URN	Identifies the LWM2M Client on one LWM2M Server (including LWM2M Bootstrap Server). Provided to the LWM2M Server during Registration, also provided to LWM2M Bootstrap Server when executing the Bootstrap procedure. Recommended URN formats are documented in Section 6.2.1 Endpoint Client Name.
LWM2M Bootstrap Server URI	URI	Uniquely identifies the LWM2M Bootstrap Server. Provided to the LWM2M Client during the Bootstrap procedure
LWM2M Server URI	URI	Uniquely identifies the LWM2M Server. Provided to the Client during Bootstrap procedure.
Short Server ID	16-bit unsigned integer	Uniquely identifies each LWM2M Server configured for the LWM2M Client. The identifier is assigned during the Bootstrap procedure. Default Short Server ID is 0 and default Short Server ID MUST NOT be used for identifying the LWM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the LWM2M Server.
Human Readable Object URN	URN for the OMA Management Object	Assigned by the Object specification.
Object ID	16-bit unsigned integer	Uniquely identifies the Object in the LWM2M Client. This identifier is assigned by OMA.
Object Instance ID	16-bit unsigned	Uniquely identifies the Object Instance of

Identifier	Semantics	Description
	integer	the Object within the LWM2M Client. This identifier is assigned by LWM2M Client or LWM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the Object Instance.
Resource ID	16-bit unsigned integer	Uniquely identifies the Resource within the Object. Short integer ID, with a range assigned by the Object specification and unique to that Object, and a Reusable Resource ID range assigned by OMA and re-usable between Objects.
Resource Instance ID	16-bit unsigned integer	Uniquely identifies the Resource Instance in the Resource. This identifier is assigned by LWM2M Client or LWM2M Server.

Table 18: LWM2M Identifiers

6.2.1 Endpoint Client Name

Following formats are RECOMMENDED for this identifier to guarantee uniqueness:

Format
UUID URN: Identify a device using a Universally Unique IDentifier (UUID). The UUID specifies a valid, hex digit character string as defined in [RFC4122]. The format of the URN is urn:uuid:########-####-####-############
OPS URN: Identify a device using the format <OUI> "-" <ProductClass> "-" <SerialNumber> as defined in Section 3.4.4 of [TR-069]. The format of the URN is urn:dev:ops:<OUI> "-" <ProductClass> "-" <SerialNumber>.
OS URN: Identify a device using the format <OUI> "-"<SerialNumber> as defined in Section 3.4.4 of [TR-069]. The format of the URN is urn:dev:os:<OUI> "-"<SerialNumber>.
IMEI URN: Identify a device using an International Mobile Equipment Identifiers [3GPP-TS_23.003]. The IMEI URN specifies a valid, 15 digit IMEI. The format of the URN is urn:imei:###############
ESN URN: Identify a device using an Electronic Serial Number. The ESN specifies a valid, 8 digit ESN. The format of the URN is urn:esn:########
MEID URN: Identify a device using a Mobile Equipment Identifier. The MEID URN specifies a valid, 14 digit MEID. The format of the URN is urn:meid:##############

Other URN formats MAY be used. In particular, URN formats defined in [DMREPPRO] Section 5.5 can be used.

6.2.2 Reusable Resources

When Objects are designed for a similar purpose, for example Objects for use in network management, or Objects for use in embedded device automation, similar Resources are useful in more than one Object. For example in embedded device automation, Objects for different purposes may contain common Resource types such as digital input, digital output, analogue input, analogue output, dimmer value, unit, min measurement, max measurement, value range etc.

If a Resource can feasibly be re-used with the same meaning in multiple Object definitions, it can be defined as a Reusable Resource ID and registered with OMNA. Other Objects may then make use of this Reusable Resource ID in another Object definition. The definition of the Resource MUST be the same with the exception of the Multiple Resource, Mandatory and Description fields.

6.3 Data Formats for Transferring Resource Information

Four data formats are defined by the LWM2M Enabler in this section: plain text, opaque, TLV and JSON.

The LWM2M Server MUST support all data formats. The LWM2M Client MUST support the TLV data format for all requests. In addition a LWM2M Client MAY choose to support other media type formats in the table below i.e., JSON, plain text and opaque.

The response message content type MUST be specified in using one of the supported Media Type.

A LWM2M Server data request MAY contain one option specifying the Content-Format the Server would prefer to receive for the payload; if this Content Format is not accepted by the LWM2M Client, the request is rejected; if the LWM2M Client doesn't support that option or the LWM2M Server expresses no data format preference, the LWM2M Client will use its own preferred data format reported in the Content Format of the response message.

The IANA registered Media Type supported in LWM2M TS 1.0 are listed in the table below

Data Format	**IANA Media Type**	**Numeric Content-Formats [CoAP]**
Plain Text	text/plain	0
Core Link Param	application/link-format	40
Opaque	application/octet-stream	42
TLV	application/vnd.oma.lwm2m+tlv	TBD
JSON	application/vnd.oma.lwm2m+json	TBD

6.3.1 Plain Text

The plain text format is used for "Read" and "Write" operations on singular Resources where the value of the Resource is simply represented as an UTF-8 encoded string. This string can contain a character sequence, integer number, decimal number or any other sequence of valid UTF-8 characters as per Appendix C.

For example a request to the example client's Device Object Instance, Manufacturer Resource would return the following plain text payload:

```
Req: GET /3/0/0

Res: 2.05 Content
Open Mobile Alliance
```

This data format has a Media Type of text/plain

6.3.2 Opaque

The opaque format is used for "Read" and "Write" operations on singular Resources where the value of the Resource is an opaque sequence of binary octets. This data format is used for binary Resources such as firmware images or application specific binary formats.

This data format has a Media Type of application/octet-stream

6.3.3 TLV

For “Read” and “Write” operations, the binary TLV (Type-Length-Value) format is used to represent an array of values or a singular value using a compact binary representation, which is easy to process on simple embedded devices. The format has a minimum overhead per value of just 2 bytes and a maximum overhead of 5 bytes depending on the type of Identifier and length of the value. The maximum size of an Object Instance or Resource in this format is 16.7 MB. The format is self-describing, thus a parser can skip TLVs for which the Resource is not known.

This data format has a Media Type of application/vnd.oma.lwm2m+tlv

The format is an array of the following byte sequence, where each array entry represents an Object Instance, Resource, or Resource Instance:

Field	Format and Length	Description
Type	8-bits masked field: 0bxxxxxxxx (MSB is the bit following 0b) Bit numbering is 0 for the LSB to 7 for the MSB	Bits 7-6: Indicates the type of Identifier. 00= Object Instance in which case the Value contains one or more Resource TLVs 01= Resource Instance with Value for use within a multiple Resource TLV 10= multiple Resource, in which case the Value contains one or more Resource Instance TLVs 11= Resource with Value
		Bit 5: Indicates the Length of the Identifier. 0=The Identifier field of this TLV is 8 bits long 1=The Identifier field of this TLV is 16 bits long
		Bit 4-3: Indicates the type of Length. 00=No length field, the value immediately follows the Identifier field in is of the length indicated by Bits 2-0 of this field 01 = The Length field is 8-bits and Bits 2-0 MUST be ignored 10 = The Length field is 16-bits and Bits 2-0 MUST be ignored 11 = The Length field is 24-bits and Bits 2-0 MUST be ignored
		Bits 2-0: A 3-bit unsigned integer indicating the Length of the Value.
Identifier	8-bit or 16-bit unsigned integer as indicated by the Type field.	The Object Instance, Resource, or Resource Instance ID as indicated by the Type field.
Length	0-24-bit unsigned integer as indicated by the Type field.	The Length of the following field in bytes.
Value	Sequence of bytes of Length	Value of the tag. The format of the value depends on the Resource's data type (See Appendix C).

Table 19: TLV format and description

Each TLV entry starts with a Type byte that indicates if the TLV contains an Object Instance, a Resource, multiple Resources, or a Resource Instance. Object Instance and Resource with Resource Instance TLVs contains other TLVs in their value. The hierarchy is as follows and may be up to 3 levels deep. The Object Instance TLV is only required if multiple Object Instances are returned in a request.

- Object Instance TLV, which contains
    - Resource TLVs or
    - multiple Resource TLVs, which contains

- Resource Instance TLVs

The following figure illustrates the possible nesting of Object Instance, Resource, multiple Resources, and Resource Instance TLVs. One or several Resource TLVs, and/or one or several multiple Resource TLVs MAY be nested in an Object Instance TLV. A multiple Resource TLV contains one or several Resource Instance TLVs.

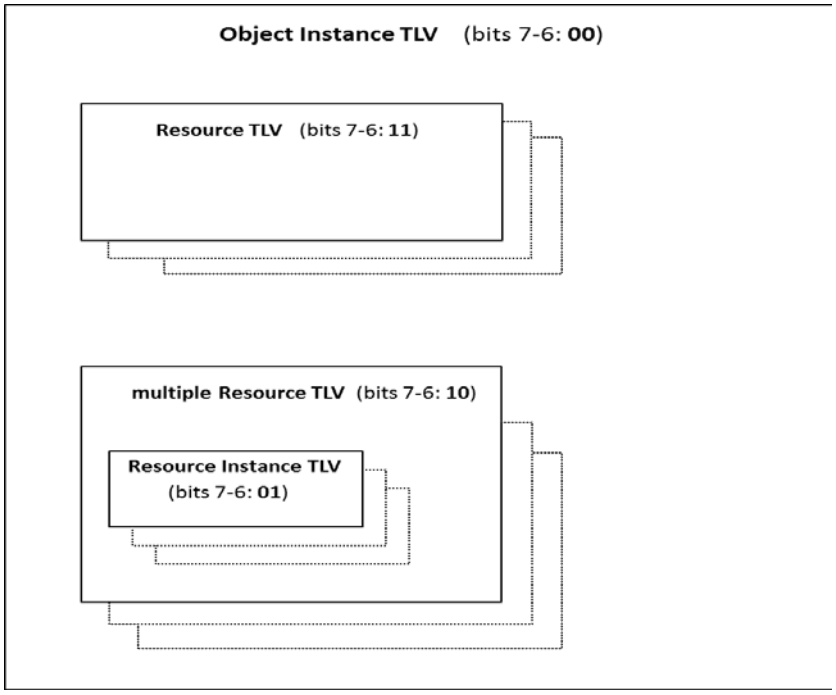


Figure 15: TLV nesting

6.3.3.1 Single Object Instance Request Example

In this example, a request for the Device Object Instance of the LWM2M example client is made (GET /3/0). The client responds with a TLV payload including all of the readable Resources. This TLV payload would have the following format. The total payload size with the TLV encoding is 121 bytes.

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Manufacturer Resource	0b11 0 01 000	0x00	0x14 (20 bytes)	Open Mobile Alliance [String]	23
Model Number	0b11 0 01 000	0x01	0x16 (22 bytes)	“Lightweight M2M Client” [String]	25
Serial Number	0b11 0 01 000	0x02	0x09 (9 bytes)	“345000123” [String]	12
Firmware Version	0b11 0 00 011	0x03	(3 bytes)	“1.0” [String]	5
Available Power Sources	0b10 0 00 110	0x06	(6 byte)	The next two rows	2
Available Power Sources[0]	0b01 0 00 001	0x00	(1 byte)	0X01 [8-bit Integer]	3
Available Power Sources[1]	0b01 0 00 001	0x01	(1 byte)	0X05 [8-bit Integer]	3

Power Source Voltage	0b10 0 01 000	0x07	0x08 (8 bytes)	The next two rows	3
Power Source Voltage[0]	0b01 0 00 010	0x00	(2 bytes)	0X0ED8 [16-bit Integer]	4
Power Source Voltage[1]	0b01 0 00 010	0x01	(2 bytes)	0X1388 [16-bit Integer]	4
Power Source Current	0b10 0 00 111	0x08	(7 bytes)	The next two rows	2
Power Source Current[0]	0b01 0 00 001	0x00	(1 byte)	0X7D [8-bit Integer]	3
Power Source Current[1]	0b01 0 00 010	0x01	(2 bytes)	0X0384 [16-bit Integer]	4
Battery Level	0b11 0 00 001	0x09	(1 byte)	0x64 [8-bit Integer]	3
Memory Free	0b11 0 00 001	0x0A	(1 byte)	0x0F [8-bit Integer]	3
Error Code	0b10 0 00 011	0x0B	(3 byte)	The next row	2
Error Code[0]	0b01 0 00 001	0x00	(1 byte)	0x00 [8-bit Integer]	3
Current Time	0b11 0 00 100	0x0D	(4 byte)	0x5182428F [32-bit Integer]	6
Time Zone	0b11 0 00 110	0x0E	(6 byte)	“+02:00” [String]	8
Supported Binding and Modes	0b11 0 00 001	0x0F	(1byte)	“U” [String]	3
				Total	121

6.3.3.2 Multiple Object Instance Request Example

In this example, a request for both the Access Control Objects of the LWM2M example client is made (GET /2). The client responds with a TLV payload including both Object Instances (0 and 1) and their Resources. This TLV payload would have the following format. The total payload size with the TLV encoding is 40 bytes.

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Access Control Object Instance 0	**0b00 0 01 000**	**0x00**	**(17 bytes)**	**The next 6 rows**	**3**
Object ID	0b11 0 00 001	0x00	(1 byte)	0x03 [8-bit Integer]	3
Object Instance ID	0b11 0 00 001	0x01	(1 byte)	0x01 [8-bit Integer]	3
ACL	0b10 0 00 110	0x02	(6 bytes)	The next 2 rows	2
ACL [1]	*0b01 0 00 001*	*0x01*	*(1 byte)*	*0b11 10 0000*	*3*
ACL [2]	*0b01 0 00 001*	*0x02*	*(1 byte)*	*0b10 00 0000*	*3*
Access Control Owner	0b11 0 00 001	0x03	(1 byte)	0x01 [8-bit Integer]	3

Access Control Object Instance 1	**0b00 0 01 000**	**0x01**	**(17 bytes)**	**The next 6 rows**	**3**
Object ID	0b11 0 00 001	0x00	(1 byte)	0x04 [8-bit Integer]	3
Object Instance ID	0b11 0 00 001	0x01	(1 byte)	0x02 [8-bit Integer]	3
ACL	0b10 0 00 110	0x02	(6 bytes)	The next 2 rows	2
ACL [1]	*0b01 0 00 001*	*0x01*	*(1 byte)*	*0b10 00 0000*	*3*
ACL [2]	*0b01 0 00 001*	*0x02*	*(1 byte)*	*0b10 00 0000*	*3*
Access Control Owner	0b11 0 00 001	0x03	(1 byte)	0x01 [8-bit Integer]	3
				Total	40

6.3.3.3 Example of Request on an Object Instance containing an Object Link Resource

Examples are based on the LWM2M Object Tree illustration of Figure 26. The TLV format doesn't report Object hierarchy.

Example 1) request to Object A: Get /A/0

TLV	**Type Byte**	**ID Byte(s)**	**Length Byte(s)**	**Value**	**Total Bytes**
Ress 0 lnk	**0b1 0 0 01000**	**0x00**	**0x0C(12 bytes)**	**The next 2 rows**	**3**
Ress 0 lnk [0]	0b11 000 100	0x00	(4bytes)	0x000B 0000 (ObjID: ObjInstID) Link to Object B, Instance 0	6
Ress 0 lnk [1]	0b11 0 00 100	0x01	(4 byte)	0x000B 0001	6
Ress 1	**0b11 0 01 000**	**0x01**	0x0D[**String]**	"8613800755500"	**16**
Ress 2	**0b11 0 00 100**	**0x02**	**Integer (4bytes)**	**xxxxxxxx**	**6**
				Total	37

Example 2) request to Object B: Get /B: TLV payload will contain 2 Object Instances

TLV	**Type Byte**	**ID Byte(s)**	**Length Byte(s)**	**Value**	**Total Bytes**
Object B Instance 0	**0b00 0 01 000**	**0x00**	**0x23 (35 bytes)**	**The next 3 rows**	3
Ress 0	**0b11 0 01 000**	0x00	0x0B	"myService 1"	14
Ress 1	**0b11 0 01 000**	0x01	0x0F	"Internet.15.234"	15
Ress 2 lnk	0b11 0 00 100	*0x02*	*(4bytes)*	0x000C 0000 (ObjID: ObjInstID) Link to Object C, Instance 0	*6*
Object B Instance 1	**0b00 0 01000**	**0x01**	**0x23 (35 bytes)**	**The next 3 rows**	**3**
Ress 0	0b11 0 01 000	0x00	0x0B	"myService 2"	14

Ress 1	0b11 0 00 000	0x01	0x0F	“Internet.15.235”	15
Ress 2 lnk	0b11 0 00 100	0x02	(4bytes)	MAX_ID MAX_ID(no link)	6
				Total	76

6.3.4 JSON

When a LWM2M Client is supporting the JSON data format and such a format is used to transport Object Instance(s), multiple resource and single resource values for both “Read” and “Write” operations, JSON payload MUST use the format defined in this section. Such a format MAY be used for transporting a single value of a Resource

The format MUST comply to [SENML] JSON representation extended for supporting LWM2M Object Link data type and MUST support all attributes defined in Table 20.

According to [SENML] semantics, JSON data format in LWM2M, is composed of optional attributes (Base Time, Base Name) and of a mandatory Resource Array having one or more entries. Each array entry contains several optional or mandatory parameters (Name, Time...).

Each entry of the JSON format is a Resource Instance, where the name need to be prepended by the optional Base Name attribute to form the unique identifier of this Resource instance.

- When the Base Name is absent, the URI of the request is used as the default value for the Base Name
- When the Base Name is present, the name of the entry has to be modulated accordingly to still uniquely identify the resource instance

Note: In both cases, the name of this array entry is a URI path relative to the Base Name which could simply be the request URI when Base Name is absent.

The JSON is useful for transporting multiple Resource Instances for example when transporting all Instances of an Object with all Resources, and Resource Instances within a single LWM2M Client response.

In particular, when Base Name is set to the LWM2M Object root (e.g “/”), the JSON format may support to return a hierarchy of Object Instances when Object Link datatype resources are reported (example given below). The resource instances tree report is performed in using a Breadth-First traversal strategy (see JSON second example below); a given Object Instance MUST appear at most once in that report. The JSON format also includes optional time fields, which allows for multiple versions of representations to be sent in the same payload. The time fields MUST only be used when sending notifications. Historical version of notifications are typically generated when “Notification Storing When Disabled or Offline” resource of LWM2M Server Object is set to true (see Appendix D.2) and when the Device comes on line after having been disabled for a period of time.

This JSON data format has a Media Type of application/vnd.oma.lwm2m+json

Attributes	JSON Variable	Mandatory?	Description
Base Name	bn	No	The base name string which is prepended to the Name value of the entry for forming a globally unique identifier for the resource.
Base Time	bt	No	The base current time which the Time values are relative to as a Time data type (See Appendix C)
Resource Array	e	Yes	The Resource list as JSON value array according to [SENML] with Array parameter extension (Object Link)
	Array Parameters		

	Name	n	Yes	The Name value is prepended by the Base Name value to form the name of the resource instance. The resulting name uniquely identifies the resource instance from all others. Example: • if Base Name is "/" , the Array entry Name of the Resource is {Object}/{Object Instance}/ {Resource}/{Resource Instance} • when Base Name is not present, the default Base Name is the request URI. i.e. if the a request URI is /{Object}/{Object Instance}, the array entry Resource name will be {Resource}/{Resource Instance}
	Time	t	No	The time of the representation relative to the Base Current Time in seconds (a negative integer) for a notification. Required only for historical representations.
	Float Value	v	One value field is mandatory	Value as a JSON float if the Resource data type is Integer, Float, or Time.
	Boolean Value	bv		Value as a JSON Boolean if the Resource data type is boolean.
	ObjectLink Value	ov		Value as a JSON string if the Resource data type is Objlnk Format according to Appendix C (e.g "10:03")
	String Value	sv		Value as a JSON string for all other Resource data types. If the Resource data type is opaque the string value holds the Base64 encoded representation of the Resource.

Table 20: JSON format and description

For example a request to Device Object of the LWM2M example client (Get /3/0) would return the following JSON payload. This example has a size of 444 bytes.

```
{"e":[
  {"n":"0","sv":"Open Mobile Alliance"},
  {"n":"1","sv":"Lightweight M2M Client"},
  {"n":"2","sv":"345000123"},
  {"n":"3","sv":"1.0"},
  {"n":"6/0","v":1},
  {"n":"6/1","v":5},
  {"n":"7/0","v":3800},
  {"n":"7/1","v":5000},
  {"n":"8/0","v":125},
  {"n":"8/1","v":900},
  {"n":"9","v":100},
  {"n":"10","v":15},
  {"n":"11/0","v":0},
  {"n":"13","v":1367491215},
  {"n":"14","sv":"+02:00"},
  {"n":"15","sv":"U"}]
}
```

For example a notification about a Resource containing multiple historical representations of a Temperature Resource in the example could result in the following JSON payload:

```
{“e”:[
  {"n":"1/2","v":22.4,"t":-5},
  {"n":"1/2","v":22.9,"t":-30},
  {"n":"1/2","v":24.1,"t":-50}],
 "bt":25462634
}
```

For example a request to Object A of the LWM2M example from Figure 26 (Get /A/0) would return the following JSON payload.

Because the Base Name is specified, the full hierarchy linked to the Instance 0 of Object A can be reported in a single response (Object B Instance 0 & 1, and Instance 0 of Object C are part of the payload). This example has a size of 435 bytes.

```
{ "bn":"/",
  "e":[
  {"n":"A/0/0/0","ov":"B:0"},
  {"n":"A/0/0/1","ov":"B:1"},
  {"n":"A/0/1","sv":"8613800755500"},
  {"n":"A/0/2","v":1},
  {"n":"B/0/0","sv":"myService1"},
  {"n":"B/0/1","sv":"Internet.15.234"},
  {"n":"B/0/2","ov":"C:0"},
  {"n":"B/1/0","sv":"myService2"},
  {"n":"B/1/1","sv":"Internet.15.235"},
  {"n":"B/1/2","ov":"FFFF:FFFF"},
  {"n":"C/0/0","sv":"85.76.76.84"},
  {"n":"C/0/1","sv":"85.76.255.255"}]
}
```

7. Security

The LWM2M protocol is based on [CoAP] principles and utilizes the UDP and SMS transport channel bindings of the protocol. The LWM2M protocol utilizes the security mechanisms of these channel bindings to implement authentication, confidentiality, and data integrity features of the protocol between communicating LWM2M entities.

For authentication of communicating LWM2M entities, the LWM2M protocol requires that all communication between LWM2M Clients and LWM2M Servers as well as LWM2M Clients and LWM2M Bootstrap Servers are authenticated using mutual authentication. This means that a:

- LWM2M Client MUST authenticate a LWM2M Server prior to exchange of any information.
- LWM2M Server MUST authenticate a LWM2M Client prior to exchange of any information.
- LWM2M Client MUST authenticate a LWM2M Bootstrap Server prior to exchange of any information.
- LWM2M Bootstrap Server MUST authenticate a LWM2M Client prior to exchange of any information.

For confidentiality and data integrity of information between communicating LWM2M entities, the LWM2M protocol requires that all communication between LWM2M Clients and LWM2M Servers as well as LWM2M Clients and LWM2M Bootstrap Servers are encrypted and integrity protected. This means that a:

- LWM2M Client MUST encrypt and integrity protect data communicated to a LWM2M Server.
- LWM2M Server MUST encrypt and integrity protect data communicated to a LWM2M Client.
- LWM2M Client MUST encrypt and integrity protect data communicated to a LWM2M Bootstrap Server.
- LWM2M Bootstrap Server MUST encrypt and integrity protect data communicated to a LWM2M Client.

The LWM2M protocol specifies that authorization of LWM2M Servers to access Object Instances and Resources within the LWM2M Client is provided through Access Control Object Instances within the LWM2M Client.

7.1 UDP Channel Security

The UDP channel security for [COAP] is defined by the Datagram Transport Layer Security (DTLS) [RFC6347], which is the equivalent of TLS v1.2 [RFC5246] for HTTP and utilizes a subset of the Cipher Suites defined in TLS. (Refers to TLS Cipher Suite registry http://www.iana.org/assignments/tls-parameters/tls-parameters.xml)

The DTLS binding for CoAP is defined in Section 9 of [CoAP]. DTLS is a long-lived session based security solution for UDP. It provides a secure handshake with session key generation, mutual authentication, data integrity and confidentiality.

Since the LWM2M protocol utilizes DTLS for authentication, data integrity and confidentiality purposes, the LWM2M Client and LWM2M Server SHOULD keep a DTLS session in use for as long a period as can be safely achieved without risking compromise to the session keys and counters. If a session persists across sleep cycles, encrypted and integrity-protected storage SHOULD be used for the session keys and counters.

Note that the Client-Server relationship of DTLS (i.e., who initiated the handshake) is separate from the Client-Server relationship of LWM2M.

Considering that any device with a LWM2M Client can be managed by any LWM2M Server and LWM2M Bootstrap Server the choice of Cipher Suites is not limited to the list defined in Section 9 of [CoAP]. Due the sensitive nature of Bootstrap Information, a particular care has to be taken to ensure protection of that data inducing constraints and dependencies within LWM2M Client/ Bootstrap Server relationship according to the adopted security mode.

Concerning Bootstrap from Smartcard, the same care has to be taken and a secure channel between the Smartcard and the LWM2M Device SHOULD be established as described in Appendix G in reference to [GLOBALPLATFORM 3], [GP SCP03].

The keying material used to secure the exchange of information using DTLS session is obtained using one of the bootstrap modes defined in Section 5.2.2 Bootstrap Modes. The formats of the keying material carried in the LWM2M Security Object Instances are defined in Appendix E.1.1.

The Resources (i.e., “Security Mode”, “Public Key or Identity”, “Server Public Key or Identity” and “Secret Key”) in the LWM2M Security Object that are associated with the keying material are used either

1) for providing UDP channel security in “Client Registration”, “Device Management & Service Enablement”, and “Information Reporting” Interfaces if the LWM2M Security Object Instance relates to a LWM2M Server, or,
2) for providing channel security in Bootstrap Interface if the LWM2M Security Object instance relates to a LWM2M Bootstrap Server.

LWM2M Clients MUST either be directly provisioned for use with a target LWM2M Server (Manufacturer Pre-configuration bootstrap mode) or else be provisioned for secure bootstrapping with an LWM2M Bootstrap Server. Any LWM2M Client which supports Client or Server initiated bootstrap mode MUST support at least one of the following secure methods:

1) Bootstrapping with a strong (high-entropy) pre-shared secret, as described in 7.1.1. The cipher-suites defined in this section MUST NOT be used with only a low-entropy pre-shared secret. The LWM2M Client MUST use a unique pre-shared secret, one which is unique to each LWM2M Client.
2) Bootstrapping with a temporary, low-entropy pre-shared secret (such as a PIN, password and private serial number) using the cipher-suite TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256, as defined in RFC5489. The LWM2M Client MUST use a unique pre-shared secret, one which is unique to each LWM2M Client. The LWM2M Client and LWM2M Bootstrap Server MUST discard this temporary secret after first usage, and MUST not allow it to be re-used for subsequent bootstrapping.
3) Bootstrapping with a public key or certificate-based method (as described in 7.1.2 and 7.1.3). The LWM2M Client MUST use a unique key-pair, one which is unique to each LWM2M Client.

For full interoperability, a LWM2M Bootstrap Server MUST support all of these methods.

NOTE: The above security methods can also be used by the LWM2M Bootstrap Server to provision KIc and KID for the SMS Secured Packet Structure mode (see Section 7.2.2 for SMS Secured Packet Structure mode).

7.1.1 Pre-Shared Keys

A LWM2M server MUST support the Pre-Shared Key mode of DTLS with the Cipher Suites below:

- TLS_PSK_WITH_AES_128_CCM_8 [RFC6655] as defined in Section 9.1.3.1 of [CoAP]
- TLS_PSK_WITH_AES_128_CBC_SHA256 as defined in [RFC5487]

A LWM2M Client MUST support the Pre-Shared Key mode of DTLS with at least one of the Cipher Suites specified for the LWM2M Server. The LWM2M Client MUST use the value of the "Public Key or Identity" Resource for "PSK identity" in [RFC4279] and the value of "Secret Key" Resource for "PSK" in [RFC4279] as defined in Appendix E.1.

The LWM2M Client and LWM2M Server MAY support the use of other Cipher Suites.

For all Cipher Suites using AES in an LWM2M implementation, the hashing functions SHOULD be SHA256.

For all Cipher Suites using AES in an LWM2M implementation, the hashing functions MUST NOT be SHA-1, and MUST NOT be MD5, and MUST NOT be any other hashing function that is weaker than SHA-1 and MD5 or otherwise deprecated.

A LWM2M Client negotiates with the LWM2M Server the best method during the DTLS handshake for establishing the DTLS session.

This security mode is appropriate for LWM2M deployments where there is an existing trust relationship between the LWM2M Server and Client. The same PSKs and PSK IDs need to be generated, and installed on the Client and Server. When using a Bootstrap Server, this security mode requires a 3-way trust relationship between the Bootstrap Server, LWM2M Server(s) and LWM2M Client(s): namely Bootstrap Server got the secret key (PSK) from Server(s), and should also share a pre-provisioned secret with Client(s) for ensuring secure DTLS Bootstrap communication.

Using Smartcard PSK provisioning needs no pre-existing trust relationship between LWM2M Server(s) and LWM2M Client(s). The pre-established trust relationship is simply between the LWM2M Server(s) and the SmartCard(s).

Notes: Some pre-cautions using TLS_PSK_WITH_AES_128_CBC_SHA256

Security wise, there are issues with CBC:

(1) Prior to TLS 1.1 IV selection is broken (solution: use TLS 1.1 or higher, known work-around for earlier version: record splitting)

(2) Implementing authenticated decryption (checking padding and mac) without any side channel is pretty hard (see Lucky 13 and its numerous variants) - known fix: the encrypt-then-mac extension (RFC 7366).

7.1.2 Raw Public Key Certificates

If a LWM2M Server supports Raw Public Key Certificates it MUST support the Cipher Suites below:

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 as defined in Section 9.1.3.2 of [CoAP]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in [RFC5289]

If a LWM2M Client supports Raw Public Key Certificates it MUST support at least one of the Cipher Suites supported by the LWM2M Server.

The LWM2M Client MUST use the value of the "Public Key or Identity" Resource for its Raw Public Key certificate and the value of "Secret Key" Resource for its Private Key as defined in Appendix E.1.

If the LWM2M Client and LWM2M Server supports Raw Public Key Certificates, they MAY support the use of other Cipher Suites.

If the LWM2M Client or LWM2M Server supports ECDHE and ECDSA for Raw Public Key Certificates, SHA-1 MUST NOT be used, and MD5 MUST NOT be used, and any other hashing function that is weaker than SHA-1 and MD5 or otherwise deprecated MUST NOT be used. The minimum key length MUST be at least 256 bits.

This security mode is appropriate for LWM2M deployments where there is an existing trust relationship between the LWM2M Server and Client. When using a Bootstrap Server, this security mode requires a 3-way trust relationship between the Bootstrap Server, LWM2M Server(s) and LWM2M Client(s): namely Bootstrap Server got the Client private key from Server(s), and should also share a pre-provisioned secret with Client(s) for ensuring secure DTLS Bootstrap communication.

The LWM2M Client MUST use the value of the "Public Key or Identity" Resource for its Raw Public Key certificate and the value of the "Secret Key" Resource for its Private Key as defined in Appendix E.1. The LWM2M Client MUST also use the "Server Public Key or Identity Resource" to determine the expected value of the LWM2M Server's raw public key, and MUST check that the raw public key presented by the LWM2M server exactly matches this stored public key.

Similarly, the LWM2M Server MUST store its own private and public keys, and MUST have a stored copy of the expected client public key. The server MUST check that the raw public key presented by the LWM2M client exactly matches this stored public key.

The server and client MUST use different key-pairs, and the LWM2M client MUST use a unique key-pair, one which is unique to each LWM2M client.

Using Smartcard RPK certificates provisioning needs no pre-existing trust relationship between LWM2M Server(s) and LWM2M Client(s). The pre-established trust relationship is simply between the LWM2M Server(s) and the SmartCard(s).

7.1.3 X.509 Certificates

The X.509 Certificate mode requires the use of X.509v3 Certificates [RFC5280].

Certificates used in LWM2M SHOULD be signed by a root certificate, either by a public root CA or a private root.

The LWM2M Client MUST either directly trust the server's X509 certificate or trust it indirectly by verifying it is correctly signed by a trusted CA.
In the case of direct trust, the LWM2M Client MUST have a copy of the expected LWM2M server certificate stored in the corresponding “Server Public Key or Identity Resource” and MUST check that the certificate presented by the LWM2M server exactly matches this stored certificate.

In the case of indirect trust, the LWM2M Client MUST have a copy of the expected CA certificate and expected LWM2M Server Subject and/or SubjectAltName names stored in the corresponding “Server Public Key or Identity Resource”. The LWM2M Client MUST check that the certificate presented by the LWM2M server is correctly signed by the expected CA, and that it contains the expected Subject and/or SubjectAltName infomation. A LWM2M Server Certificate SHOULD include Subject and/or SubjectAltName fields listing its known DNS names and IP addresses which are included in the LWM2M Server URI Resource of the LWM2M Security Object Instance. The LWM2M Server MAY use a wild card certificate for the DNS with the host represented as an * and the rest of the domain fully qualified (e.g., *.acme.com). A wildcard with only a top level domain is not permitted (e.g., *.com). The LWM2M Client MUST check that these fields of the Certificate match the URI used to register with the LWM2M Server.

Similarly, the LWM2M Server MUST either directly trust the LWM2M Client's X509 certificate or trust it indirectly by verifying it is correctly signed by a trusted CA certificate. In the case of direct trust, the server MUST store a copy of the expected LWM2M client certificate and MUST check that the certificate presented by the LWM2M client exactly matches this stored certificate. In the case of indirect trust, the server MUST store a copy of the expected CA certificate and expected LWM2M Client Subject and/or SubjectAltName names. The server MUST check that the certificate presented by the LWM2M Client is correctly signed by the expected CA certificate, is within its stated validity period, and contains the expected Subject and/or SubjectAltName information. A LWM2M Client Certificate MUST include the Endpoint Name parameter used to register the device in the Subject Common Name (CN) field of the Certificate. Upon registration, the LWM2M Server MUST check that this CN field matches the Endpoint Name parameter of the registration message during authentication and MUST respond “4.00 Bad Request” to the LWM2M Client if these fields do not match.

If a LWM2M server supports X.509 Certificate mode it MUST support the Cipher Suites below:

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 as defined in Section 9.1.3.3 of [CoAP].
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in [RFC5289]

If a LWM2M Client supports X.509 Certificate mode it MUST support at least one of the Cipher Suites supported by the LWM2M Server. The LWM2M Client MUST use the value of the "Public Key or Identity" Resource for its X.509 certificate and the value of "Secret Key" Resource for its Private Key as defined in Appendix E.1.

If the LWM2M Client and LWM2M Server supports X.509 Certificate mode, they MAY support the use of other Cipher Suites.

If the LWM2M Client or LWM2M Server supports ECDHE and ECDSA for X.509 Certificate mode, SHA-1 MUST NOT be used, and MD5 MUST NOT be used, and any other hashing function that is weaker than SHA-1 and MD5 or otherwise deprecated MUST NOT be used. The minimum key length MUST be at least 256 bits.

This security mode does not require a pre-existing trust relationship (if all entities used X.509 certificate security mode) between the LWM2M Client and LWM2M Server, nor between a LWM2M Bootstrap Server and a LWM2M Client. However, in the case of indirect trust, all entities need a trust relationship with the CA(s) that issued the certificates used in LWM2M Servers and Clients.

Using Smartcard with certificates provisioning needs no pre-existing trust relationship between LWM2M Server(s) and LWM2M Client(s). The pre-established trust relationship is simply between the LWM2M Server(s) and the SmartCard(s).

A LWM2M Client SHOULD wait until it has accurate absolute time before contacting the LWM2M Server or LWM2M Bootstrap Server. If the LWM2M Client uses direct trust, and has no accurate absolute time, it MUST ignore those components of the LWM2M Server or LWM2M Bootstrap Server certificate that involve absolute time, e.g. not-valid-before and not-valid-after certificate restrictions. If the LWM2M Client uses indirect trust, and has no accurate absolute time, it MUST otherwise establish that the LWM2M Server or LWM2M Bootstrap Server certificate is within its validity period. For example, the LWM2M Client can just know the date (or year), and the server certificate can have a long validity period, extending well before and after the expected time period needed to bootstrap the device.

7.1.4 “NoSec” mode

It is highly recommended to always use LWM2M with one of the security mechanisms described above. However, there are few scenarios and use cases where security is provided by lower layers. For example LWM2M devices in a controlled environment behind a gateway, or, tests focussing first on other functions before performing end-to-end tests including security.

7.2 SMS Channel Security

Channel security for [COAP] has been defined for the UDP transport and is based on the Datagram Transport Layer Security (DTLS) [RFC6347]

This section defines the security modes for the transport of COAP over SMS.

LWM2M Clients supporting SMS, when the SMS Channel is only used for debugging purposes MAY support the NoSec mode.

LWM2M Clients supporting UDP and SMS, when the SMS Channel is only used for triggering as defined in chapter 8.4 MUST support the adequate mechanism for securing UDP Channel as defined in chapter 7.1 UDP channel security. Those

clients MAY use any SMS security mode. In particular SMS NoSec mode can be used for SMS triggering since all other communication will be secured by UDP channel security.

Using SMS NoSec for SMS triggering could induce issues as “Denial of Service” (DoS) , SMS auto reply attacks (based on PoR :) and is strongly not recommanded

LWM2M Clients supporting SMS for communications other than triggering, or supporting only the SMS Channel MUST support SMS Secured Mode. In any security mode except for debugging purposes, when an SMS message is received from an MSISDN that is not recorded in the LWM2M Server SMS Number resource of the LWM2M Server Access Security, the SMS message MUST be silently ignored.

7.2.1 SMS “NoSec” mode

It is highly recommended to always use LWM2M with one of the security mechanisms described in this section. However, there are few scenarios and use cases where security is provided by lower layers. For example LWM2M devices in a controlled environment behind a gateway, or, tests focussing first on other functions before performing end-to-end tests including security.

This security profile is also useful to support SMS triggering when all other exchanges run over UDP Channel.

7.2.2 SMS Secured mode

The SMS Secured mode specified in this section MUST be supported when the SMS binding is used.

A LWM2M Client which uses the SMS binding MUST either be directly provisioned for use with a target LWM2M Server (Factory Bootstrap or Bootstrap from Smartcard) or else be able to bootstrap via the UDP binding.

The end-point for the SMS channel (delivery of mobile terminated SMS, and sending of mobile originated SMS) MAY be either on the Smartcard or on the Device. When the LWM2M Client device doesn’t support a Smartcard, the end-point is on the LWM2M Client device.

A LWM2M Client, Server or Bootstrap Server supporting SMS binding MUST discard SMS messages which are not correctly protected using the expected parameters stored in the “SMS Binding Key Parameters” Resource and the expected keys stored in the “SMS Binding Secret Keys” Resource, and MUST NOT respond with an error message secured using the correct parameters and keys.

7.2.2.1 Device end-point

The Secured Packet Structure is based on [3GPP TS 31 115]/[ETSI TS 102 225] which was originally designed for securing packet structures for UICC based applications. However, for LWM2M it is suitable for securing the SMS payload exchanged between client and server. Usage of Secured Packet Structure Packet mode in LWM2M device needs evolution towards the introduction of a secure environment. The intention is to evolve the specifications in the next LWM2M release.

In LWM2M Enabler 1.0 if the SMS channel end-point is on the Device, the Channel security for [COAP] is based on the Datagram Transport Layer Security (DTLS) [RFC6347]. For that reason the main lines of section 7.1 on “UDP Channel Security” relative to DTLS binding on CoAP are also applicable to that section.

This sub-section describes how to bind CoAP/DTLS message to the SMS channel and specifies the restrictions on DTLS for fitting the SMS channel specific functioning and narrow bandwidth [SMS-DTLS]

7.2.2.1.1 DTLS Handshake considerations

DTLS Handshake Phase requires the exchanges of several logical messages (“flights”) between a Client and a Server. DTLS defines a special mechanism in order to fragment a single flight in several pieces for the emission and to reassemble the pieces to recover the original flight during reception.

However each “flight” has to be considered as monolithic, meaning if an error occurs on the exchange of one single fragment, the full flight has to be re-transmitted.

These DTLS Handshake feature leads to 2 rules for the SMS channel media:

- the 3GPP Concatenated short messages mechanism MUST NOT be used during handshake to avoid redundant overhead
- before starting the handshake phase, the DTLS implementation MUST be explicitly configured with an PMTU of 140 Bytes

7.2.2.1.2 DTLS Message Segmentation and Re-Assembly Consideration

Due to DLTS high sensibility to packet loss and following the recommendation of [SMS_DTLS], the SMS Channel media in LWM2M requires to follow the 2 rules below:

- the 3GPP Concatenated Short Message mechanisms MUST NOT be used
- the same PMTU setting used during the DTLS Hanshake phase must be kept

7.2.2.1.3 Multiplexing Security Association

This functionality specified in [SMS_DTLS] could authorize to address multiple LWM2M Clients in the same devices, each Clients having a specific identifier, carried by an extra header (7bytes) based on WAP User Datagram Protocol specification [WAP-WDP]. This functionality – DTLS sidelines - required to substract additional 7 bytes (WDP header) from the SMS effective payload and is not supported in LWM2M release 1.0. Later version of OMA LWM2M could support it through a new SMS DTLS mode (DLTS mode with support to Multiplexing Security Associations), and managing a header of 7 bytes in addition to the one specified in section 7.2.2.1.6.

7.2.2.1.4 DTLS supported authentication modes considerations

The X.509 certificate-based authentication (used in Certificate mode CoAP) exacerbates the number of fragments composing the flights needed to complete the handshake phase, and then increases the likelihood to incur packet loss. As DTLS timeout and retransmission logics apply to a given flight as a whole and not on individual fragment of it, a loss or a delay of a single fragment may disrupt the current flight, which has to be entirely retransmitted. For that reason, only PSK-based authentication MUST be supported on SMS Channel using DTLS.

7.2.2.1.5 Timers values for DTLS

The timeout is defined by retransmission timeout (RTO) in DTLS. Every unsuccessful attempt would double the interval of timer from initial timeout from RTO till a hardcoded value (60 seconds in DTLS) is crossed. The timer value should be positioned beyond SMS delivery timing, in order to achieve best efficient results for DTLS over SMS.

The suggestion is to keep the initial RTO at 10 seconds for DTLS. The attempts would be made after 10, 20, 40, 80 seconds before the value crosses the hardcoded limit (60 seconds). In total, the overall timing comes to 150 seconds (2.5 minutes) which is a fair value within which SMS would be delivered.

When SMS delivery report function is activated, reception of an SMS-STATUS-REPORT message has not to be interpreted as an indication that a previously sent handshake message, has been acted by the receiver.
Therefore, the SMS-STATUS-REPORT message MUST NOT be considerate by the DTLS timeout and retransmission function.
In order to avoid persisting messages in the network, the SMS validity period carried by the handshake messages MUST have a value higher or at least equal to the DTLS retransmission timeout (RTO)

7.2.2.1.6 Header Definitions (for one SMS)

a) SMS Frame for basic Request/Response Interaction message (no Token field required)

<table>
<tr><td colspan="4">TPDU (140 bytes)</td></tr>
<tr><td colspan="3">DTLS (29 bytes)</td><td colspan="2">CoAP + Effective Payload</td></tr>
<tr><td>Header (13)</td><td>Nonce (8)</td><td>ICV (8)</td><td colspan="2"></td></tr>
<tr><td colspan="3"></td><td>CoAP (4 bytes)</td><td>Effective Payload (107 bytes)</td></tr>
</table>

Model calculation using these header definitions,

- Overall TPDU : 140 Bytes
    - DTLS takes 29 bytes: 13 bytes (reference, RFC 6347) of header + 16 bytes of integrity check for CoAP in DTLS [RFC 6655]. Cipher suite mandated by CoAP (AES-128)
    - CoAP header 4 [CoAP]
    - Available bytes for the effective LWM2M Payload from one SMS: 107bytes

b) SMS Frame for messages of the Information Reporting Interface (Token field required)

TPDU (140 bytes)				
DTLS (29 bytes)			CoAP + Effective Payload	
Header (13)	Nonce (8)	ICV (8)		
			CoAP (4 + 8 bytes)	Effective Payload (99 bytes)

Model calculation using these header definitions,

- DTLS takes 29 bytes: 13 bytes (reference, RFC 6347) of header + 16 bytes of integrity check for CoAP in DTLS [RFC 6655] . Cipher suite mandated by CoAP (AES-128)
- CoAP header 4+8 [CoAP] (Token field required)
- Available bytes for the effective LWM2M Payload from one SMS: 99 bytes

7.2.2.2 Smartcard end-point

If the SMS channel end-point is on the smart card, a CoAP message as defined in [CoAP] MUST be encapsulated in [3GPP 31.115] Secured Packets, in implementing - for SMS Point to Point (SMS_PP) - the general [ETSI 102 225] specification for UICC based applications

The following settings MUST be applied:

Class 2 SMS as specified in [3GPP TS 23.038]. The [3GPP TS 23.040] SMS header MUST be defined as below:

- TP-PID : 111111 (USIM Data Download) as specified in [3GPP TS 23.040]
- TP-OA : the TP-OA (originating address as defined in [3GPP 23.040] of an incoming command packet (e.g CoAP request) MUST be re-used as the TP-DA of the outgoing packet (e.g CoAP response)

7.2.2.2.1 Secure SMS Transfer to UICC

A SMS Secured Packet encapsulating a CoAP request received by the LWM2M device, MUST be – according to [ETSI TS 102 225]/[3GPP TS 31.115] - addressed to the LWM2M UICC Application in the Smartcard where it will be decrypted, aggregated if needed, and checked for integrity.

If decryption and integrity verification succeed, the message contained in the SMS MUST be provided to the LWM2M Client.

If decryption or integrity verification failed, SMS MUST be discarded.

The mechanism for providing the decrypted CoAP Request to the LWM2M Client relies on basic GET_DATA commands of [GP SCP03] .This data MUST follow the format as below:

data_rcv _ ::= <address> <coap_msg>

address ::= TP_OA ; originated addresss

coap_msg ::= COAP_TAG <coap_request_length> <coap_request>

coap_request_length ::= 16BITS_VALUE

coap_request ::= CoAP message payload

NOTE: In current LWM2M release, the way the LWM2M Client Application is triggered for retrieving the available message from the Smartcard is device specific: i.e a middle class LWM2M Device implementing [ETSI TS 102 223] ToolKit with class “e” and “k” support could be automatically triggered by Toolkit mechanisms, whereas a simpler LWM2M device could rely on a polling mechanisms on Smartcard for fetching data when available.

7.2.2.2.2 Secured SMS Transfer to LWM2M Server

For sending a CoAP message to the LWM2M Server, the LWM2M Client prepares a data containing the right TP-DA to use, concatenated with the CoAP message and MUST provide that data to the LWM2M UICC Application in using the [GP SCP03] STORE-DATA command.

According to [ETSI TS 102 225] / [3GPP TS 31.115] the Smartcard will be in charge to prepare (encryption / concatenation) the CoAP message before sending it as a SMS Secure Packet ([ETSI TS 102 223] SEND_SMS command).

The SMS Secured Packet MUST be formatted as Secured Data specified in section 7.2.2.3.

The Secure Channel as specified in Annex H of this document SHOULD be used to provide the prepared data to the Smartcard.

7.2.2.3 SMS Secured Packet Binding for CoAP messages

In SMS Secured Packet Structure mode, a CoAP message as defined in [CoAP] MUST be encapsulated in [3GPP 31.115] Secured Packets, in implementing - for SMS Point to Point (SMS_PP) - the general [ETSI 102 225] specification for UICC based applications.

- The “Command Packet” command specified in [3GPP 31.115] /[ETSI TS 102 225] MUST be used for both CoAP Request and Response message
- The Structure of the Command Packet contained in the Short Message MUST follow [3GPP 31.115] specification
- SPI MUST be set as follow (see coding of SPI in [ETSI TS 102 225] section 5.2.1):
    - use of cryptographic checksum
    - use of ciphering
        - The ciphering and crypto graphic checksum MUST use either AES or Triple DES
        - Single DES MUST NOT be used
        - AES SHOULD be used
        - When Triple DES is used , then it MUST be used in outer CBC mode and 3 different keys MUST be used
        - When AES is used it MUST be used with CBC mode for ciphering (see coding of KIc in [ETSI TS 102 225] section 5.2.2) and in CMAC mode for integrity (see coding of KID in [ETSI TS 102 225] section 5.2.3).
    - process if and only if counter value is higher than the value in the RE
    - PoR depends on LWM2M Server Policy
- TAR MUST be set to ‘B2 02 03’ value for the LWM2M UICC Application as registered in [ETSI TS 101 220] Appendix D
- Secured Data : contains the Secured Application Message which MUST be coded as a BER-TLV, the Tag (TBD : e.g 0x05) will indicate the type (e.g CoAP type) of that message

7.3 Access Control

As the LWM2M Client MAY support one or more LWM2M Servers, there is a need to determine which operation on a given Object Instance is authorized for which LWM2M Server: Access Control Object is designed for supporting that capability. In the particular case where a single LWM2M Server Account exists in the LWM2M Client, the Server MUST have full access right on all the Objects and Object Instances in the LWM2M Client, and the Access Control Object MAY be not instantiated.

The section 7.3.1 and its sub-sections specify what MUST be applied in multiple LWM2M Servers environment. For consistency and for reducing the efforts of the LWM2M Client when switching from single to multiple LWM2M Servers environment after deployment, section 7.3.1 and its sub-sections MUST also be applicable in single LWM2M Server environment when Access Control Object is instantiated in that context.

7.3.1 Access Control Object

7.3.1.1 Access Control Object overview

Access Control Object MUST be instantiated with the following rules:

- During the Bootstrap Phase, an Access Control Object Instance MUST be assigned per Object, for identifying which Server will be authorized to instantiate this Object later. The Bootstrap Server is the owner of these Access Control Object Instances created during this phase.

- A unique Access Control Object Instance MUST be assigned per Object Instance (see Figure 16), for registering which operations can be performed by a given LWM2M Server on this associated Object Instance.
- Each Access Control Object Instance MUST only be managed by the Access Control Owner Server of that Object Instance via the Device Management and Service Enablement Interface. Within the Access Control Object Instance is an ACL Resource which MAY have zero or several Instances (see Figure 16):
  - Each ACL Resource Instance is associated to a different LWM2M Server and represents an access right determining which operations a LWM2M Server can perform on the Object Instance.
  - For a given Access Control Object Instance, the Access Control Owner LWM2M Server which doesn't have ACL Resource Instance, have full access right on Object Instance it refers to.
  - For a simple association between an ACL Resource Instance and a given LWM2M Server, the Short Server ID is assigned to ACL Resource Instance ID (see Figure 16).
  - A default ACL Resource Instance MAY be used to grant access rights to LWM2M Servers which doesn't have its own ACL Resource Instance. ID of this default ACL Resource Instance MUST be 0.
- Access Control Object is described in Appendix E.3 and Examples of Access Control Object Instances are presented in Appendix F.

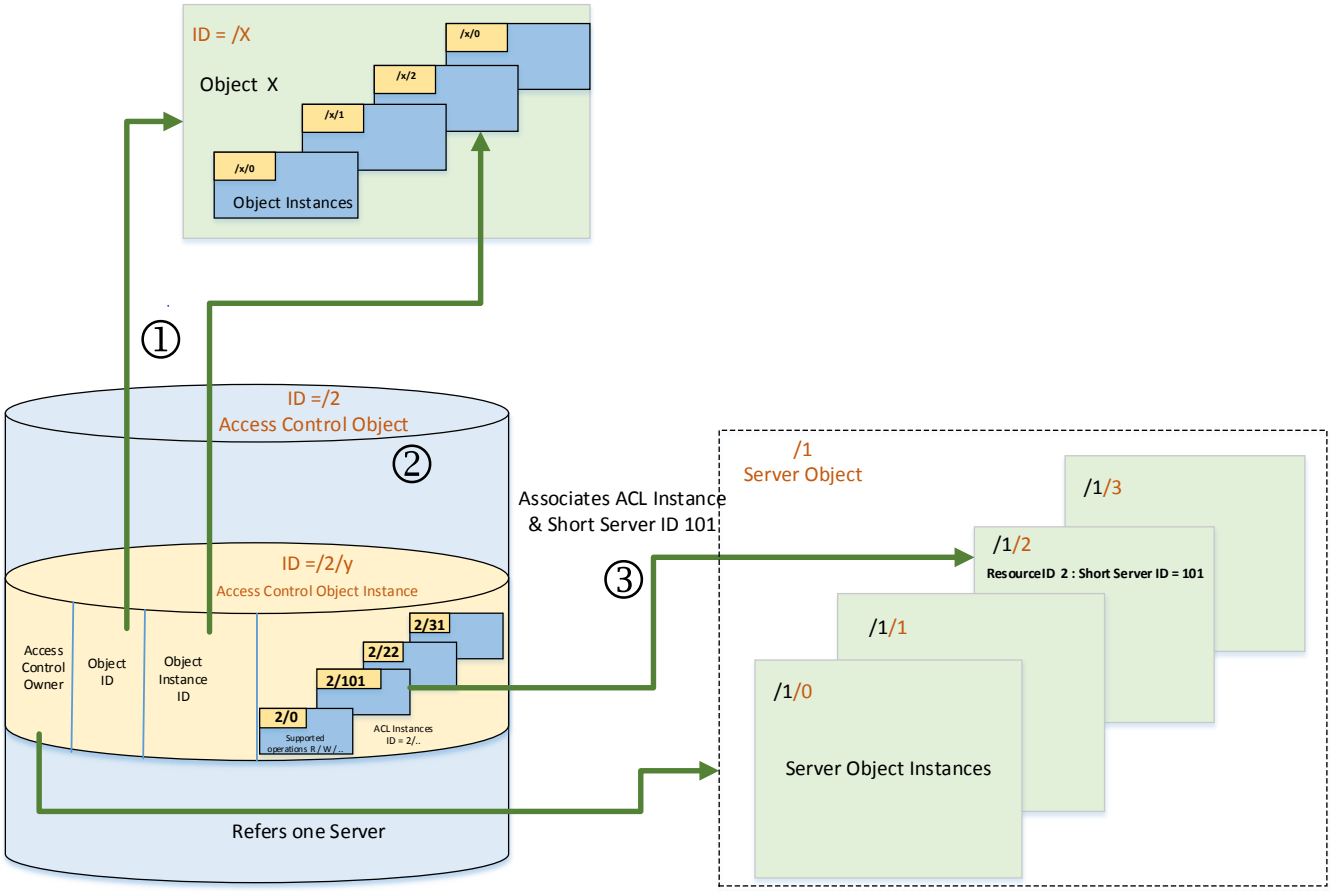


Figure 16: Illustration of the relations between the LWM2M Access Control Object and the other LWM2M Objects

7.3.1.2 Access Control Object Management

7.3.1.2.1 Access Control Object Instantiation

Access Control Object MUST be instantiated by the LWM2M Client under two circumstances:

- During Bootstrap for specifying which Server is authorized to instantiate ("Create" operation) which Object.
- When a LWM2M Server sends "Create" operation.

1. Bootstrap case

During Bootstrap, for each Object supported by the LWM2M Client, an Access Control Object Instance per Object SHOULD be created by the LWM2M Client. This Access Control Object Instance MUST be managed via the Bootstrap Interface only. The Resources value of the Access Control Object Instance MUST be set as follows:

Resource Name	Resource ID	Value
Object ID	0	ID of the targeted Object
Object Instance ID	1	MAX_ID=65535 (irrelevant)
ACL	2	A Resource Instance per LWM2M Server authorized to instantiate the Object 4th lsb: "Create" is only configured
Access Control Owner	3	MAX_ID=65535 (meaning : managed by Bootstrap Interface)

2. "Create" operation case

When a LWM2M Server creates under authorization an Object Instance (see section 7.3.2 Authorization) in the LWM2M Client, the LWM2M Client MUST instantiate an Access Control Object Instance with the following Resource values. The Access Control Owner Resource is configured with Short Server ID of the LWM2M Server.

Resource Name	Resource ID	Value
Object ID	0	ID of the targeted Object
Object Instance ID	1	ID of the newly created Object Instance
ACL	2	Any combinaison of the Access Right {none,R,W,E,D} is acceptable (Annex E.3)
Access Control Owner	3	The Short Server ID of the LWM2M Server sending "Create" operation

7.3.1.2.2 Access Control Object update

There are several cases in which a given Access Control Object Instance can be updated:

- When the LWM2M Server which is the "Access Control Owner" adds or modifies (using "Write" operation) access right on the Object Instance for a given LWM2M Server,
  a. first an ACL Resource having the targeted Short Server ID as ACL Resource Instance ID, has to be instantiated by the LWM2M Client if ACL Resource Instance for the LWM2M Server doesn't exist yet
  b. second the appropriate access right (R,W,D,E) for that targeted Server on the Object Instance has to be set as ACL Resource Instance value
- When an Object Instance is removed via "Delete" operation performed by the LWM2M Server which is the "Access Control Owner", the associated Access Control Object Instance MUST be removed by the LWM2M Client.

7.3.2 Authorization

The LWM2M Client MUST authorize a CREATE operation requested by a LWM2M Server for instantiating an Object, only if the associated Access Control Object Instance managed by the Bootstrap Interface, contains an ACL Resource Instance for that LWM2M Server set with the Access Right "Create".

The LWM2M Client MUST authorize other operations than CREATE requested by a LWM2M Server either on an Object Instance, or on Resource after performing a two-steps check:

- 1st step: the LWM2M Client gets the access right of the targeted Object Instance (as described in section 7.3.2.1) - and checks whether this access right is sufficient – according to the following table - to perform the LWM2M Server requested operation.

LWM2M Operations	Minimum Access Right
READ – OBSERVE	R
WRITE	W
DISCOVER – WRITE ATTRIBUTES	-
DELETE	D
EXECUTE	E

- 2nd step: if at step 1, the Server is granted to perform the operation the LWM2M Client needs to check if the LWM2M Server requested operation is supported by the targeted Resource or Object Instance (details are described in section 7.3.2.2, 7.3.2.3, and 7.3.2.4).

The LWM2M Object specification defines which operations are allowed to be performed on Resource within an Object Instance (Refer to Supported Operations in Appendix D LWM2M Object Template and Guidelines). The operations allowed on a given Resource MUST apply to all the Resource Instances of that Resource.

The LWM2M Client MUST support the authorization procedure described in Section 7.3.2 and its sub-sections.

7.3.2.1 Obtaining Access Right

For "Create" operation sent by the LWM2M Server, the LWM2M Client MUST get access right from the ACL Resource Instance associated to this LWM2M Server on the targeted Object, which is contained in the Access Control Object Instance provisioned during Bootstrap (Access Control Owner is MAX_ID=65535). If this access right doesn't have the "Create" value, or cannot be obtained, the LWM2M Server has no access right.

For the operations except than "Create" operation the LWM2M Client MUST perform the following procedure:

1. if this LWM2M Server is the only LWM2M Server Account declared in the LWM2M Client (ie single Server environment) , the LWM2M Server has full access right on Object Instance(s) If the LWM2M Server has more than one LWM2M Server Account, the LWM2M Client gets an Access Control Object Instance associated with the Object Instance the LWM2M Server has requested access to and MUST follow the procedure below:
   A. If the LWM2M Server is declared as Access Control Owner of this Object Instance and there is no ACL Resource Instance, then LWM2M Client gets full access right.
   B. If the Client has an ACL Resource Instance for the LWM2M Server, the LWM2M Client gets access right from that ACL Resource Instance.
   C. If the Client doesn't have ACL Resource Instance for the Server, the LWM2M Client gets access right from the default ACL Resource Instance if it exists.
   D. If the Client doesn't have default ACL Resource Instance then, the LWM2M Server has no access right, and an "Access Right Permission Denied" error code is reported to the LWM2M Server.

7.3.2.2 Operation on Resource

When the LWM2M Server targets a Resource, the LWM2M Client MUST obtain an access right for the LWM2M Server on the Object Instance that Resource belongs to according to Section 7.3.2.1 and MUST checks whether the access right is granted prior to perform the requested operation.

- If the operation is not permitted, the LWM2M Client MUST send an "Access Right Permission Denied" error code to the LWM2M Server.
- If the operation is permitted, the LWM2M Client verifies whether the Resource supports the operation.
- If the operation is not supported by the Resource, the LWM2M Client MUST send an "Operation is not supported" error code to the LWM2M Server.
- If the Resource supports the operation, the LWM2M Client MUST performs the requested operation.

7.3.2.3 Operation on Object Instance

When the LWM2M Server targets an Object Instance, the LWM2M Client MUST obtain an access right for the LWM2M Server on Object Instance according to Section 7.3.2.1 and MUST check whether the access right is granted prior to perform the requested operation.

- If the operation is not permitted, the LWM2M Client MUST send an "Access Right Permission Denied" error code to the LWM2M Server.
- If the operation is permitted, the following cases apply, according to the requested operation:
  - For the "Write" operation on an Object Instance, the LWM2M Client MUST perform the operation, if all the Resources conveyed in the operation are allowed to perform the "Write" operation. If any Resource does not support the "Write"operation, the LWM2M Client MUST inform the LWM2M Server, the Object Instance doesn't perform the requested "Write" operation by sending a "Operation is not supported" error code.
  - For the "Read" and "Observe" operations, the LWM2M Client MUST retrieve all the Resources except the Resource(s) which doesn't support the "Read" operation and sends the retrieved Resource(s) information to the LWM2M Server.
  - For the "Execute" operation, the LWM2M Client MUST NOT perform the operation, and MUST send an "Operation is not supported" error code to the LWM2M Server.
  - For the "Delete", "Write Attributes", and "Discover" operations, the LWM2M Client MUST perform the operation since those operations are not related to Resource.

7.3.2.4 Operation on Object

If a given LWM2M Server targets an Object with a "Write", "Execute", or "Delete" operation, the LWM2M Client MUST NOT perform such an operation and MUST send an "Operation is not supported" error code to the LWM2M Server.

- When the LWM2M Server targets an Object for the "Create" operation, the LWM2M Client MUST obtain an access right for the LWM2M Server on Object according to Section 7.3.2.1 "Obtaining Access Right" and MUST check whether the access right is granted prior to perform the requested operation.

  If the "Create" operation is permitted, the LWM2M Client MUST perform the instantiation on the Object only if all the mandatory Resources are specified in the "New Value" parameter (see Section 5). If all the mandatory Resources are not specified, the LWM2M Client MUST send a "Bad Request" error code to the LWM2M Server.

  Optional Resources MAY be conveyed in the "New Value" parameter as well; the LWM2M Client MAY ignore the optional resources it doesn't support. The values of the Read-only Resources MUST be setup by the LWM2M Client only. When a value is present in the "New Value" parameter, this value MUST simply be ignored. If the payload (NewValue) conveys an Object Instance ID in conflict with one already present in the LWM2M Client, the complete request MUST be rejected and a "Bad Request" error code MUST be sent back.
- The "Discover" operation on Object is specific in the sense, that no access right is needed; the LWM2M Client MUST perform the operation.

- For the “Read” and “Observe” operations, the LWM2M Client MUST obtain the access right for the LWM2M Server on each Object Instance according to Section 7.3.2.1 “Obtaining Access Right” and the LWM2M Client MUST retrieve all the Object Instances for which the LWM2M Server has “Read” access right; for each of these qualified Object Instances, the LWM2M Client MUST retrieve all the Resources except the Resources which do not support the “Read” operation. The LWM2M Client MUST then aggregate all the information individually produced by the operation on each of these Object Instances and send that to the LWM2M Server.
- For the “Write Attributes” operation, the LWM2M Client MUST perform the operation.

7.3.2.5 Notify Operation Consideration

If the LWM2M Client needs to send a “Notify” operation containing an Object Instance or a Resource to the LWM2M Server, the LWM2M Client MUST check whether the LWM2M Server is authorized for the “Read” operation. If the LWM2M Server is not authorized, the Client MUST NOT send the “Notify” operation.

8. Transport Layer Binding and Encodings

The LWM2M interfaces use the IETF Constrained Application Protocol [CoAP] as an underlying transfer protocol across IP and SMS bearers. This protocol defines the message header, request/response codes, message options and retransmission mechanisms. This section defines the subset of features from the IETF CoAP specification to be used by LWM2M interfaces.

8.1 Required Features

For realization of the LWM2M interfaces, only the basic binary CoAP message header, and a small subset of options are required. This section explicitly defines the features of the CoAP standard that are required for LWM2M.

- The 4-byte binary CoAP message header is defined in Section 3 of [CoAP]. This same base message is used for Request and Response interactions.
- Confirmable, Acknowledgement and Reset messages MUST be supported. The Reset message is used as a message layer error message in response to a malformed Confirmable message. Non-Confirmable messages MAY be used by a Client for sending Information Reporting notifications as per [Observe].
- GET, PUT, POST and DELETE methods MUST be supported. LWM2M Operations map to these methods.
- A subset of Response Codes MUST be supported for LWM2M response message mapping.
- The Uri-Path Option MUST be supported to indicate the identifier of the interface, Object Instance and Resource being requested.
- The Location-Path Option MUST be supported to indicate the handle of a registration for future update and delete operations.
- The Uri-Query Option MUST be supported.
- The Content-Type Option MUST be used to indicate the media type of the payload.
- The Accept Option MAY be included in a LWM2M Server data request, to specify the payload Content-Format this Server prefers to receive. The Client returns the preferred Content-Format if available. If this Accept option is not given or if the LWM2M Client doesn't support that option, the LWM2M Client will use its own preferred data format reported in the Content-Format of the response message. If the preferred Content-Format cannot be returned, then a 4.06 "Not Acceptable" value MUST be sent as a response.
- The Token Option MAY be used to enable multiple requests in parallel with an endpoint, and MUST be supported for the Information Reporting interface.

8.2 URI Identifier & Operation Mapping

Although CoAP supports a URI in requests, it is not used in the same way as in HTTP. The URI in CoAP is broken down into binary parts, minimizing overhead and complexity. In LWM2M only path segment and query string URI components are needed. The URI path is used to simply identify the interface, Object Instance or Resource that the request is for, and is encoded in Uri-Path options. The LWM2M Registration interface also makes use of query string parameters to pass on meta-data with the request separately from the payload. Each query parameter is encoded in a Uri-Query Option. Likewise, the LWM2M operations for each interface are mapped to CoAP Methods. All the LWM2M operations except "Notify" MUST be Confirmable CoAP message and "Notify" can be either Confirmable or Non-Confirmable CoAP message when UDP Transport Layer is used.

8.2.1 Firewall/NAT

For a firewall to support LWM2M, it should be configured to allow outgoing UDP packets to destination port 5683 (other ports can be configured), and allow incoming UDP packets back to the source address/port of the outgoing UDP packet for a period of at least 240 seconds. These UDP packets may contain DTLS or CoAP payloads. When a firewall is configured as such any LWM2M Clients behind it should use Queue Mode.

For a firewall to support LWM2M it can be configured to allow both outgoing and incoming UDP packets to destination port 5683 (other ports can be configured). These UDP packets may contain DTLS or CoAP payloads. When a firewall is configured as such any LWM2M Clients behind it are not required to use Queued Mode, but may use it for other reasons (e.g. a battery powered sleeping device).

Any LWM2M Clients behind a NAT can use Queued Mode. There are other mechanisms to transverse a NAT, however they are out of scope for the LWM2M Enabler.

8.2.2 Bootstrap Interface

The bootstrap interface is used to optionally configure a LWM2M Client so that it can successfully register with a LWM2M Server. The client bootstrap operation is performed by sending a CoAP POST request to the LWM2M Bootstrap Server at the /bs path including the Endpoint Client Name as a query string parameter. When bootstrap operation is terminated the Bootstrap Server MUST send a bootstrap finish indication

In client initiated bootstrap, when the Bootstrap Server receives Request Bootstrap operation, the Bootstrap Server performs Write and/or Delete operation. In server initiated bootstrap, the Bootstrap Server performs Write operation. The Delete operation targets an Object Instance while a Write operation targets Object, Object Instance or a Resource. The Write and Delete operation can be sent multiple times. Only in Bootstrap Interface, Delete operation MAY target to "/" URI to delete all the existing Object Instances - except LWM2M Bootstrap Server Account - in the LWM2M Client, for initialization purpose before LWM2M Bootstrap Server sends Write operation(s) to the LWM2M Client. Different from „Write" operation in Device Management and Service Enablement interface, the LWM2M Client MUST write the value included in the payload regardless of an existence of the targeting Object Instance(s) or Resource and access rights. The Bootstrap Server must send finish indication after it has sent all object instances/resources. Bootstrap Server send finish message by sending CoAP POST to "/bs" location path with empty payload

Operation	CoAP Method	URI	Success	Failure
Bootstrap Request	POST	/bs?ep={Endpoint Client Name}	2.04 Changed	4.00 Bad Request 4.15 Unsupported content format
Write	PUT	/{Object ID}/{Object Instance ID}/ {Resource ID}	2.04 Changed	4.00 Bad Request
Delete	DELETE	/{Object ID}/{Object Instance ID}	2.02 Deleted	4.00 Bad Request
Bootstrap Finish	POST	/bs	2.04 Changed	4.00 Bad Request

Table 21: Operation to Method and URI Mapping

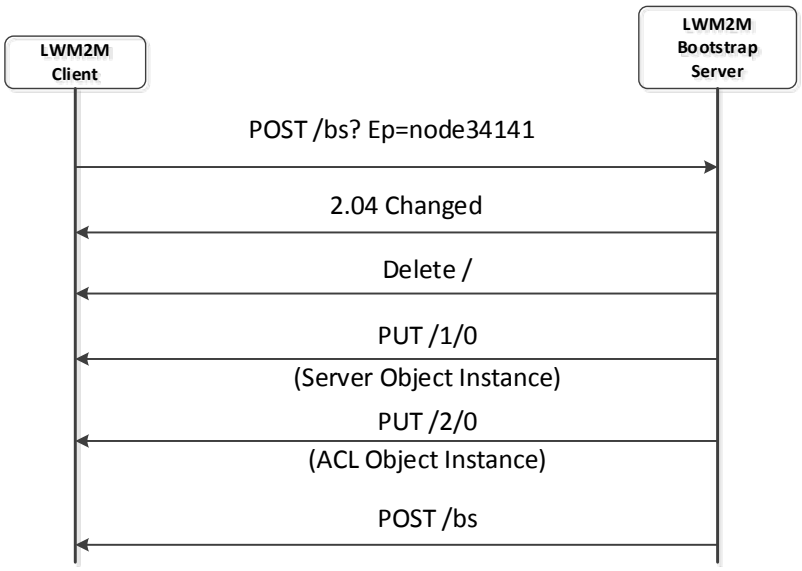


Figure 17: Example of Client initiated Bootstrap exchange.

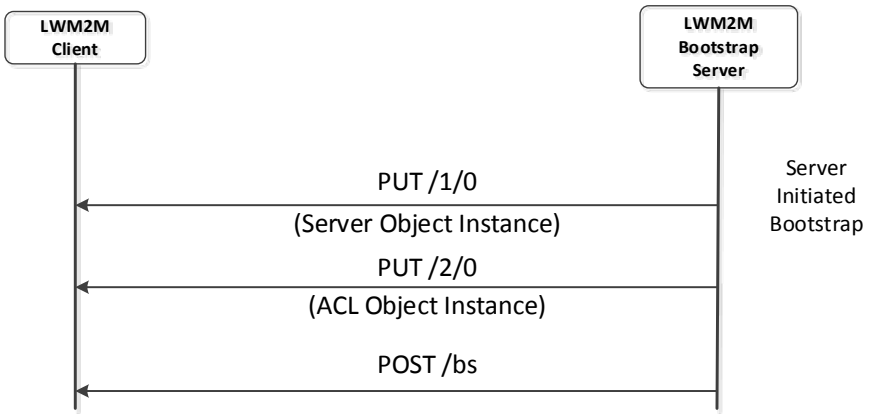


Figure 18: Example of Server initiated Bootstrap exchange.

8.2.3 Registration Interface

The registration interface is used by a LWM2M Client to register with a LWM2M Server, identified by the LWM2M Server URI. Registration is performed by sending a CoAP POST to the LWM2M Server URI, with registration parameters passed as query string parameters as per Table 22 and Object and Object Instances included in the payload as specified in Section 5.3.1. The response includes Location-Path Options, which indicate the path to use for updating or deleting the registration. The server MUST return a location under the /rd path segment.

Registration update is performed by sending a CoAP POST to the Location path returned to the LWM2M Client as a result of a successful registration.

De-registration is performed by sending a CoAP DELETE to the Location path returned to the LWM2M Client as a result of a successful registration.

Operation	CoAP Method	URI	Success	Failure
Register	POST	/rd?ep={Endpoint Client Name}&lt={Lifetime}&sms={MSISDN} &lwm2m={version}&b={binding}	2.01 Created	4.00 Bad Request, 4.03 Forbidden
Update	POST	/{location}?lt={Lifetime}&sms={MSISDN} &b={binding}	2.04 Changed	4.00 Bad Request, 4.04 Not Found
De-register	DELETE	/{location}	2.02 Deleted	4.00 Bad Request, 4.04 Not Found

Table 22: Operation to Method and URI Mapping

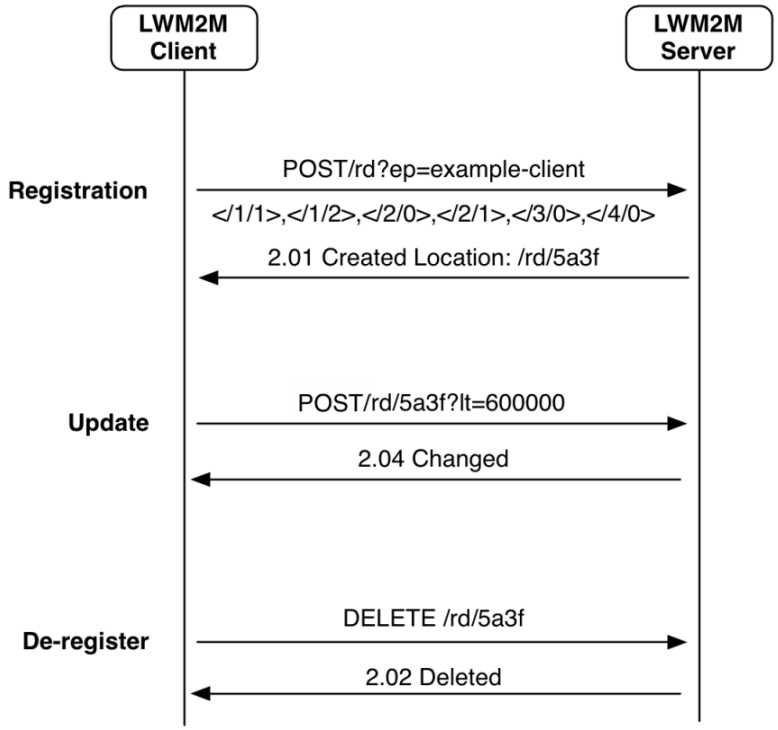


Figure 19: Example register, update and de-register operation exchanges (shorthand in [CoAP] example style, actual messages using CoAP binary headers)

8.2.4 Device Management & Service Enablement Interface

The Device Management & Service Enablement Interface is used to access Resource, an array of Resource Instances, an Object Instance or all the Object Instances of an Object. An Object Instance is identified by the path / {Object ID}/ {Object Instance ID}. If Object doesn't support multiple Object Instances, the Object Instance is identified by the path / {Object ID}/0. A Resource is identified by the path / {Object ID}/ {Object Instance ID}/ {Resource ID}.

An Object Instance or Resource is Read by sending a CoAP GET to the corresponding path. The response includes the value in the corresponding Plain Text, Opaque, TLV or JSON format according to the specified Content-Format (see section 6.3).The request MAY specify an Accept option containing the preferred Content-Format to receive. When the specified content format is not supported by the LWM2M Client, the request MUST be rejected.

An Object Instance or Resource is Written to by sending either a CoAP PUT or a COAP POST to the corresponding path. The request includes the value to be written in the corresponding Plain Text, Opaque, TLV or JSON format according to the 'Content-Format option which MUST be specified [CoAP]. The Write request MUST be rejected when the specified Content Format is not supported by the LWM2M Client

A CoAP PUT is used for the Replace and CoAP POST is used for Partial Update mechanism of the "Write" operation as described in 5.4.3.

A Resource is Executed by sending a CoAP POST to the corresponding path. The request MAY include a list of arguments as value of the payload expressed in Plain Text format. The definition of the Executable Resource and its arguments is given in Appendix D.

The list of argument can be empty, 2 arguments of the arguments list are separated by a comma. The syntax of the arguments is provided in Section Execute (5.2.3)

Note that the behaviour of the "Execute" operation, whether it uses arguments and how those are interpreted, and how it returns values is specified in the Resource description of the Object.

An Object Instance is Created by sending a CoAP POST to the corresponding path. The request includes the value to be written in the corresponding TLV or JSON format according to the Content-Format option which MUST be specified. The

rules governing the creation of Resources in the targeted Object Instance are specified in section 7.3.2.3 (Operation on Object Instance). If Object Instance is not listed at the request, the LWM2M Client MUST assign ID of that Object Instance and send back Object Instance ID with "2.01 Created" to the LWM2M Server when Object Instance is Created.

An Object Instance is Deleted by sending a CoAP DELETE to the corresponding path.

When a Resource supports multiple instances the Resource value is an array of Resource Instances.

[NOTIFICATION] class Attributes MAY be set by a LWM2M Server using the "Write Attributes" operation on the corresponding path, and can be accessed using the "Discover" operation. One or more Attributes can be written at a time. The values of these Attributes are used by the Information Reporting interface to determine how often Notifications are sent regarding that Resource. A LWM2M Client MAY support a set of these Attributes for each LWM2M Server it is configured for.

A Write Attribute command specifies which value is affected to which Attribute and at which level (Object / Object Instance / Resource) it is assigned. In a similar way, the same command without value for the specified Attribute, MUST be used to de-assign this Attribute for the given level; then the precedence rules applies when notification occurs (section 5.1.1 Attributes definitions and Rules)

As example:

a) Write Attributes /3/0/9?pmin=1 means the Battery Level value will be notified to the Server with a minimum interval of 1sec

b) Write Attributes /3/0/9?pmin means the Battery Level will be notified to the Server with a minimum value (pmin) given by the default one (resource 2 of Object Server ID=1), or with another value if this Attribute has been assigned at another level (Object or Object Instance: see section 5.1.1).

Operation	CoAP Method	Path	Success	Failure
Read	GET Accept: Content Format ID (see section 6.3)	/{Object ID}/{Object Instance ID}/{Resource ID}	2.05 Content	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable
Discover	GET Accept: application/link-format	/{Object ID}/{Object Instance ID}/{Resource ID}	2.05 Content	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Write	PUT Content Format:	/{Object ID}/{Object Instance ID}/{Resource ID}	2.04 Changed	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed, 4.06 Not Acceptable
	POST Content Format:	/{Object ID}/{Object Instance ID}		
Write Attributes	PUT	/{Object ID}/{Object Instance ID}/{Resource ID}?pmin={minimum period}&pmax={maximum period}&gt={greater than}&lt={less than}&stp={step}	2.04 Changed	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Execute	POST	/{Object ID}/{Object Instance ID}/{Resource ID}	2.04 Changed	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed
Create	POST Content Format:	/{Object ID}/{Object Instance ID}	2.01 Created	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable
Delete	DELETE	/{Object ID}/{Object Instance ID}	2.02 Deleted	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed

Table 23: Operation to Method Mapping

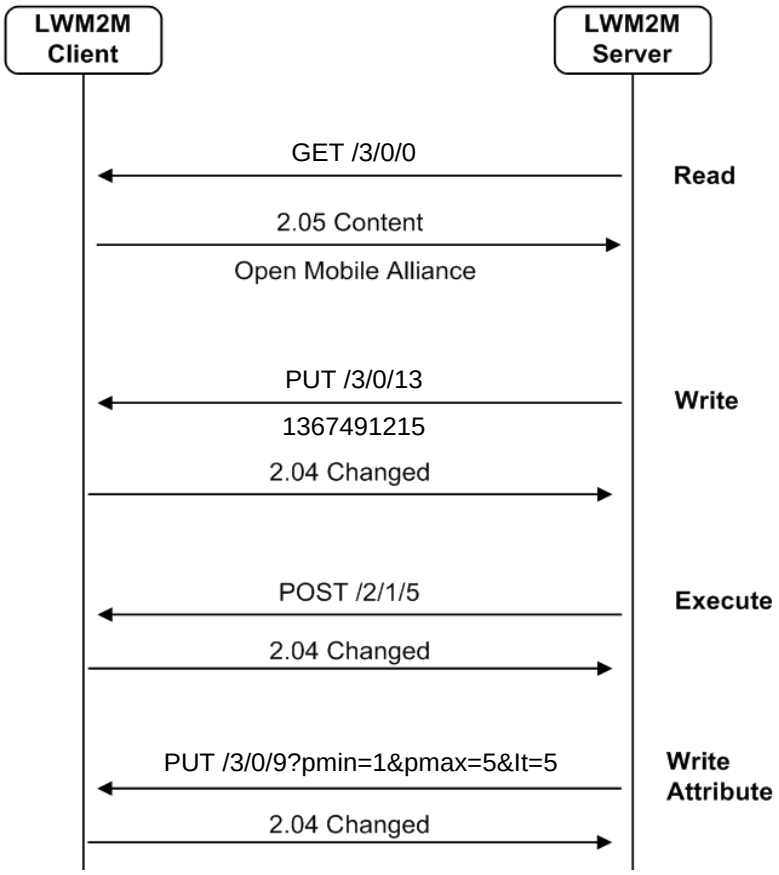


Figure 20: Example of Device Management & Service Enablement interface exchanges.

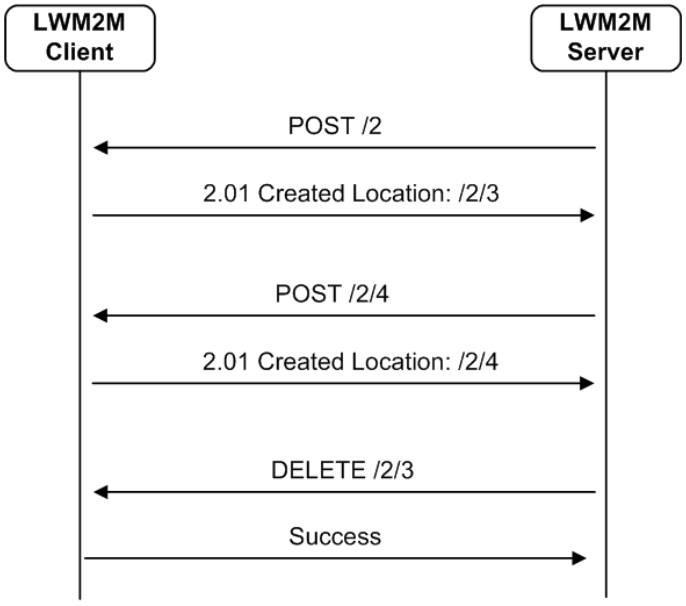


Figure 21: Example of Object Creation and Deletion.

8.2.5 Information Reporting Interface

Periodic and event-triggered reporting about Resource values from the LWM2M Client to the LWM2M Server is achieved through CoAP Observation [OBSERVE]. This simple mechanism allows the LWM2M Server to send a GET request with Observe option =0 for an Object, Object Instance, or Resource which results in asynchronous notifications whenever that Object Instance changes (periodically or as a result of an event). Token of CoAP layer is used to match the asynchronous notifications with the Observe GET. The LWM2M Server can cancel the "Observe" operation by sending Reset message as the response for Notify message in which the LWM2M Server is not interested any more. When the LWM2M Client receives a Reset in response of a "Notify" operation, the LWM2M Client MUST cancel the Observation regardless if the Notify was sent as a confirmable CoAP message as defined in [OBSERVE] or as a non-confirmable CoAP message. The LWM2M Server can also cancel the "Observe" operation at any moment, on a specified Resource, or specified Object Instance(s), by sending a GET request with Observe option=1. The LWM2M Server may set the Observe attributes of a Resource to affect the behavior its notifications using the "Write Attributes" operation (see Section 5.4.4 Write Attributes).

Operation	CoAP Method	Path	Success	Failure
Observe	GET with Observe option = 0	/{Object ID}/{Object Instance ID}/{Resource ID}	2.05 Content with Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Cancel Observation	Reset message			
	GET with Observe option = 1	/{Object ID}/{Object Instance ID}/{Resource ID}	2.05 Content without Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized,4.05 Method Not Allowed
Notify	Asynchronous Response		2.05 Content with Values	

Table 24: Operation to Method Mapping

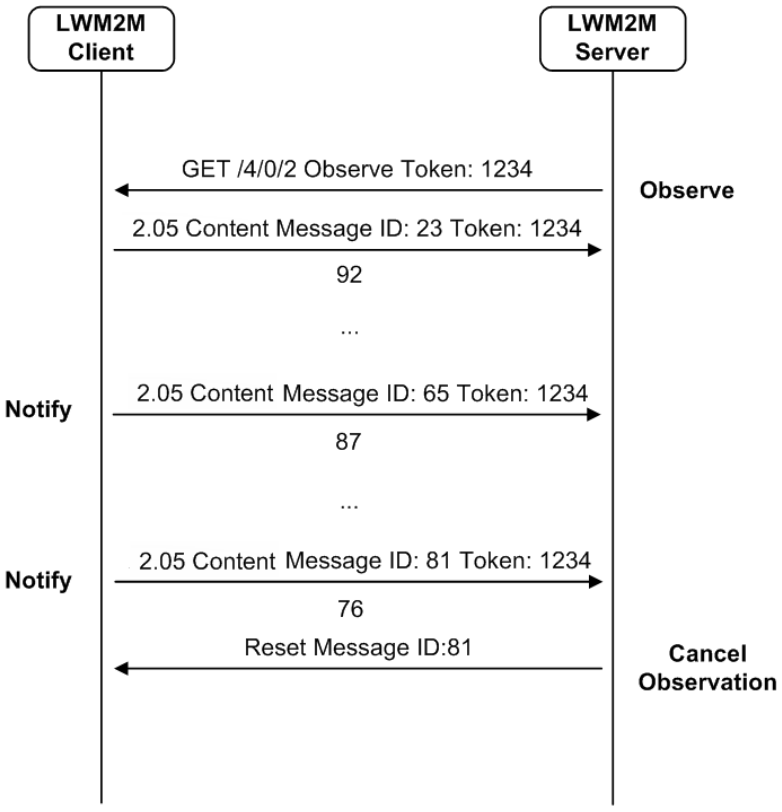


Figure 22: Example of an Information Reporting exchange.

8.3 Queue Mode Operation

The LWM2M Server MUST support Queue Mode and the LWM2M Client SHOULD support Queue Mode.

When the LWM2M Client has registered with Current Transport Binding and Mode parameter including “Q” (see chapter 5.4), The LWM2M Server does not immediately send downlink requests on the transport layer used in Queue mode, but instead waits until the LWM2M Client is online on the transport layer.

The LWM2M Client lets the LWM2M Server know it is awake by sending a registration update message as a Confirmable message. The LWM2M Server then makes any queued requests to the LWM2M Client in a serial fashion. The LWM2M Client MUST wait at least ACK_TIMEOUT [COAP] seconds from the last CoAP message it sent to the LWM2M Server before intentionally going offline. If the LWM2M Server is not successful in sending a request – i.e. Server gives up on receiving an acknowledgment or reset -, then it stops emptying the queue and keeps the request for the next time the LWM2M Client is online.

A typical Queue Mode sequence follows the following steps:

1. The LWM2M Client registers to the LWM2M Server and requests the LWM2M Server to run in Queue mode by using the correct Binding value in the registration.

2. The LWM2M Client uses the CoAP ACK_TIMEOUT parameter to set a timer for how long it shall stay awake since last sent message to the LWM2M Server. After ACK_TIMEOUT without any messages from the LWM2M Server, the LWM2M Client SHOULD sleep until sending next periodic “Update” operation.

3. When the LWM2M Server receives a message from the Client (e.g. a notification or a registration update), it checks its request queue for the LWM2M Client and performs the needed CoAP operation(s) (e.g. GET, PUT, and POST). Note: There could be several requests in the queue. Each request is sent serially to the LWM2M Client, waiting for request to be Acknowledged before sending the next request. If a request is unsuccessful then the request returns to the queue that has been previously fetched, to the same position in the queue. The LWM2M Client may have pending Observer notifications.

Below is an example flow for Queue Mode in relation to Device Management & Service Enablement Interface.

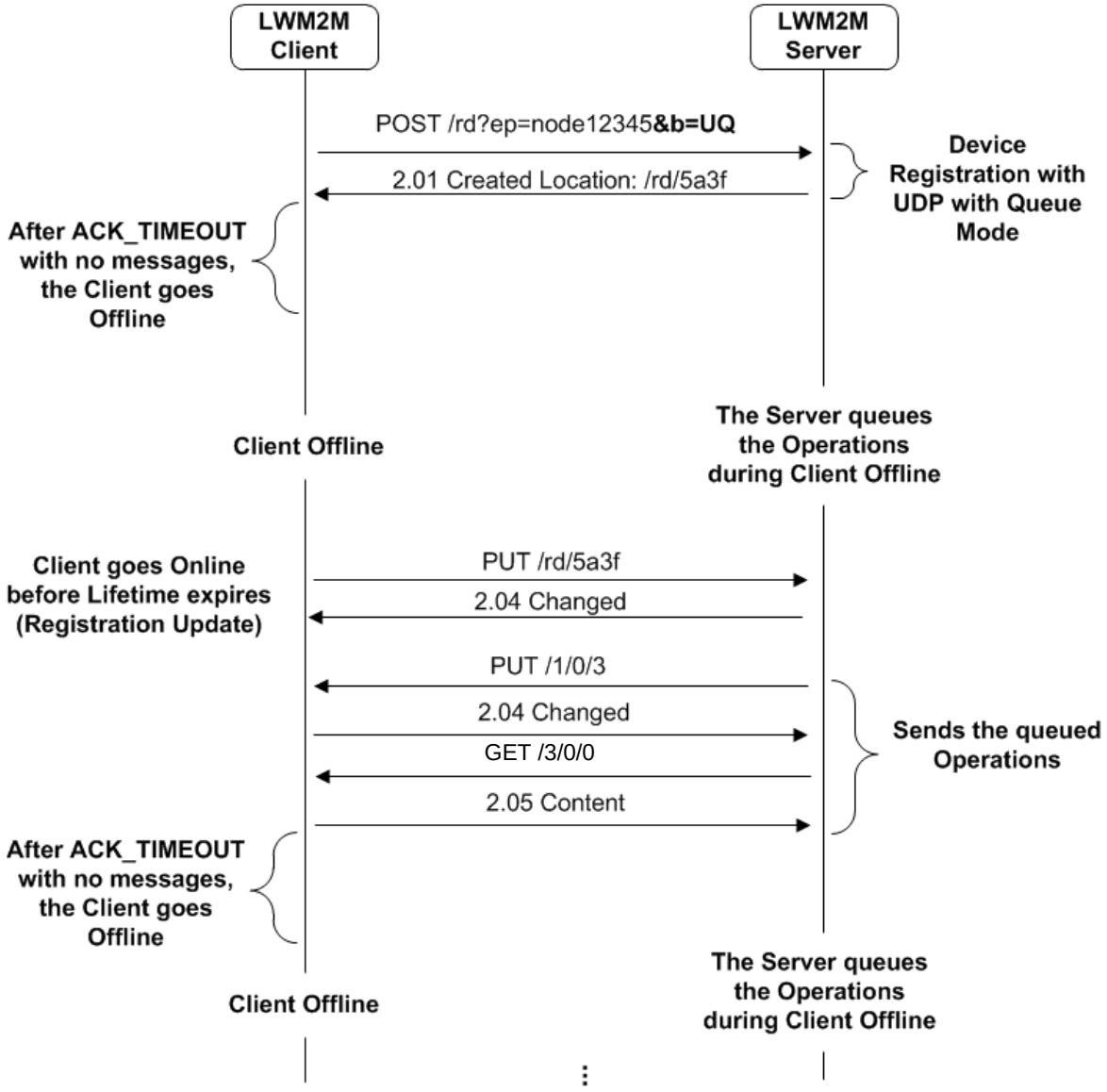


Figure 23: Example of Device Management & Service Enablement interface exchanges for Queue Mode.

Below is an example flow for Queue Mode in relation to Information Reporting Interface

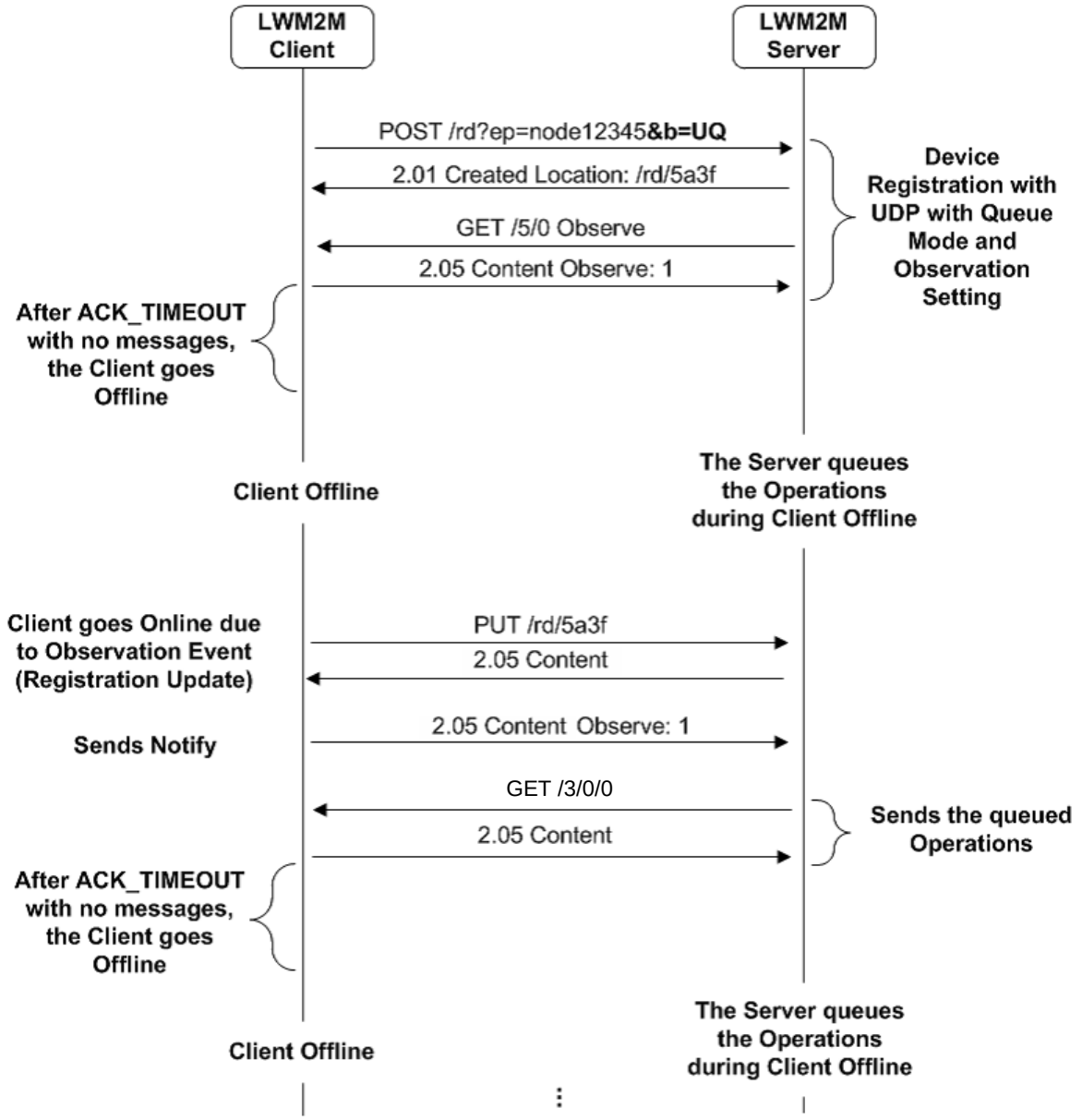


Figure 24: Example of an Information Reporting exchange for Queue Mode.

8.4 Update Trigger Mechanism

When the LWM2M Client has registered with Current Transport Binding and Mode parameter with "UQS", the LWM2M Server MAY make the LWM2M Client come online and register on UDP by executing Registration Update Trigger Resource in Device Object Instance (refer to Appendix E.2). Below is an example flow how to trigger the LWM2M Client in Queue Mode to send Update message to the LWM2M Server regardless of expiration of Lifetime. Post /1/x/8 would bring the LWM2M Client online to talk to the LWM2M server, where "x" represents the right instance pointing to the server.

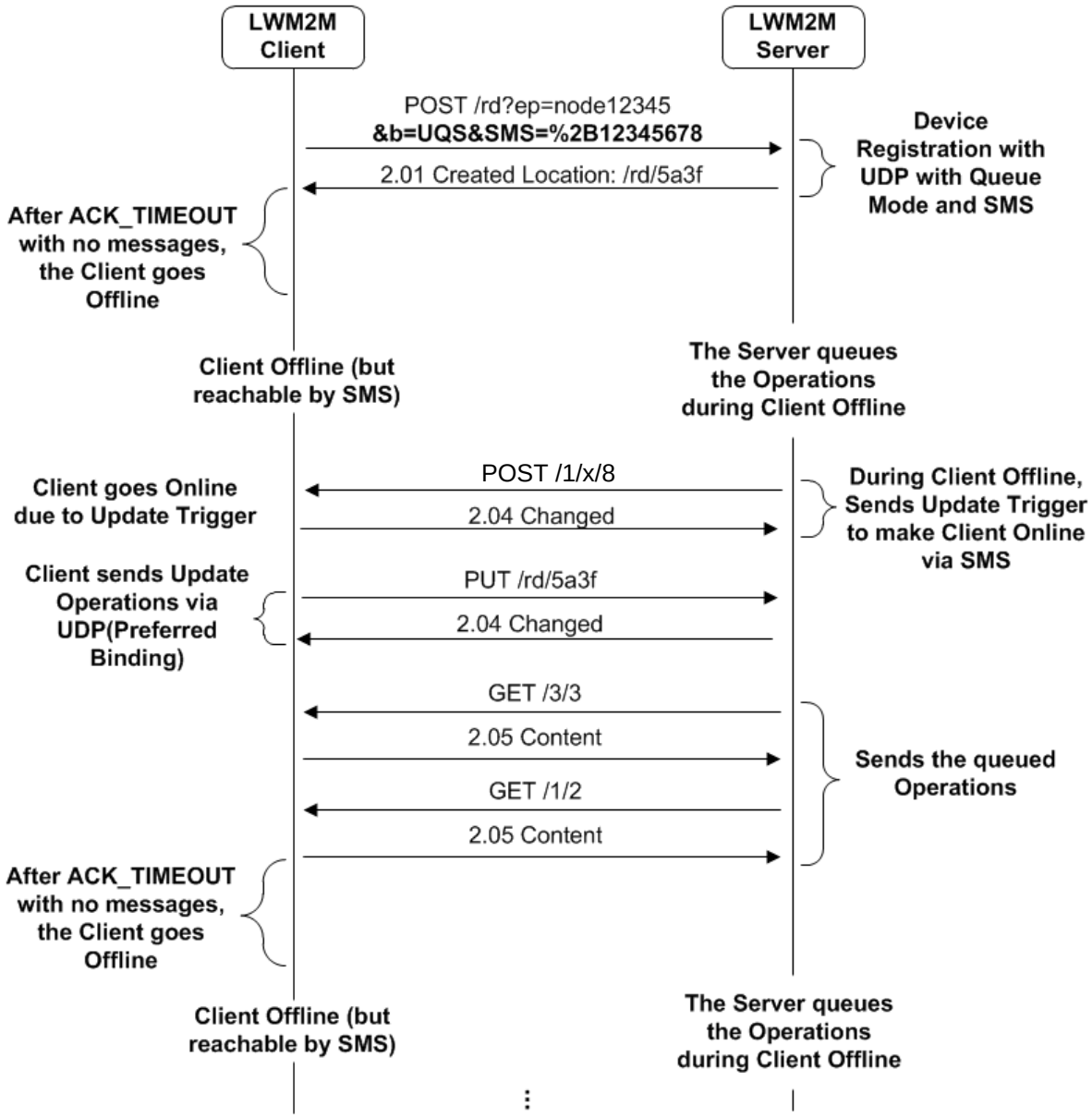


Figure 25: Example of Device Management & Service Enablement interface exchanges for Queue Mode with SMS Registration Update Trigger.

8.5 Response Codes

This section lists available response codes of each operation. The codes are divided into each interface. These are the only valid response codes defined in for the LWM2M Enabler.

Operations	Available CoAP Response Codes	Reason Phrase
Bootstrap Interface		
Bootstrap Request	2.04 Changed	Request Bootstrap is completed successfully
	4.00 Bad Request	Unknown Endpoint Client Name
Write	2.04 Changed	"Write" operation is completed successfully
	4.00 Bad Request	The format of data to be written is different
	4.15 Unsupported content format	The specified format is not supported
Delete	2.02 Deleted	"Delete" operation is completed successfully
	4.00 Bad Request	Bad or unknown URI provided
Bootstrap Finish	2.04 Changed	Bootstrap Finished is completed successfully
	4.00 Bad Request	Bad URI provided
Client Registration Interface		
Register	2.01 Created	"Register" operation is completed successfully
	4.00 Bad Request	The mandatory parameter is not specified or unknown parameter is specified Unknown Endpoint Client Name Endpoint Client Name does not match with CN field of X.509 Certificates
	4.03 Forbidden	The Endpoint Client Name registration in the LWM2M Server is not allowed.
Update	2.04 Changed	"Update" operation is completed successfully
	4.00 Bad Request	The mandatory parameter is not specified or unknown parameter is specified
	4.04 Not Found	URI of "Update" operation is not found
De-register	2.02 Deleted	"De-register" operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.04 Not Found	URI of "De-register" operation is not found
Device Management and Service Enablement Interface		
Create	2.01 Created	"Create" operation is completed successfully
	4.00 Bad Request	Target (i.e., Object) already exists Mandatory Resources are not specified Content Format is not specified
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of "Create" operation is not found
	4.05 Method Not Allowed	Target is not allowed for "Create" operation
	4.15 Unsupported content format	The specified format is not supported
Read	2.05 Content	"Read" operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of "Read" operation is not found

	4.05 Method Not Allowed	Target is not allowed for “Read” operation
	4.06 Not Acceptable	None of the preferred Content-Formats can be returned
Write	2.04 Changed	“Write” operation is completed successfully
	4.00 Bad Request	The format of data to be written is different
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of ”Write“ operation is not found
	4.05 Method Not Allowed	Target is not allowed for “Write” operation
	4.15 Unsupported content format	The specified format is not supported
Delete	2.02 Deleted	“Delete” operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of “Delete” operation is not found
	4.05 Method Not Allowed	Target is not allowed for “Delete” operation
Execute	2.04 Changed	“Execute” operation is completed successfully
	4.00 Bad Request	The LWM2M Server doesn’t understand the argument in the payload
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of “Execute” operation is not found
	4.05 Method Not Allowed	Target is not allowed for “Execute” operation
Write Attributes	2.04 Changed	“Write Attributes” operation is completed successfully
	4.00 Bad Request	The format of attribute to be written is different
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of “Write Attributes” operation is not found
	4.05 Method Not Allowed	Target is not allowed for Write Attributes operation
Discover	2.05 Content	“Discover” operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of “Discover” operation is not found
	4.05 Method Not Allowed	Target is not allowed for Discover operation
Information Reporting Interface		
Observe	2.05 Content	“Observe” operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of “Observe” operation is not found
	4.05 Method Not Allowed	Target is not allowed for “Observe” operation
	4.06 Not Acceptable	None of the preferred Content-Formats can be returned
Notify	2.05 Content	“Notify” operation is completed successfully

Table 25: Response Codes

If any operation in table 21, 24 and 25 cannot be completed in the client and the reason cannot be described by a more specific response code, then a generic response code of "**5.00 Internal Server Error"** MUST be returned.

8.6 Transport Bindings

The LWM2M Server and the LWM2M Client MUST support UDP binding specified in Section 8.6.1 UDP Binding and the LWM2M Server SHOULD support SMS binding and the LWM2M Client MAY support SMS binding specified in Section 8.6.2 SMS Binding.

8.6.1 UDP Binding

The CoAP binding for UDP is defined in [CoAP]. The protocol has a IANA registered scheme of coap:// and a default port of 5683. The UDP binding is used in NoSec (no security) mode. Reliability over the UDP transport is provided by the built-in retransmission mechanism of CoAP.

8.6.2 SMS Binding

CoAP is used over SMS in this transport binding by placing a CoAP message in the SMS payload using 8-bit encoding. SMS concatenation MAY be used for messages larger than 140 characters. CoAP retransmission is disabled for this binding. An LWM2M Client indicates the use of this binding by including a parameter (sms) in its registration to the LWM2M Server including the node's MSISDN number. The LWM2M Client MAY interact with the server using both UDP and SMS bindings.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-LightweightM2M-V1_0	04 Sep 2012	All	TS baseline agreed as in OMA-DM-LightweightM2M-2012-0078-INP_TS_kick_off
	18 Sep 2012	6, 7	Incorporates input to committee: OMA-DM-LightweightM2M-2012-0083R01-CR_Skeleton_Base_Line OMA-DM-LightweightM2M-2012-0090R02-CR_TS_Resource_Model OMA-DM-LightweightM2M-2012-0061R04-CR_Interfaces
	24 Oct 2012	6, 7, Appendix A	OMA-DM-LightweightM2M-2012-0095R01- CR_TS_Interface_and_Resource_Additions
	30 Oct 2012	7, 8	OMA-DM-LightweightM2M-2012-0097R01- CR_Identifiers_and_Security_Considerations
	17 Nov 2012	2, 6, 7, 8, 9, 10	OMA-DM-LightweightM2M-2012-0088R04-CR_Transfer_Protocol OMA-DM-LightweightM2M-2012-0098R02- CR_Bootstrap_Information_and_Modes OMA-DM-LightweightM2M-2012-0099R01-CR_Default_ACL_Entry OMA-DM-LightweightM2M-2012-0100R02- CR_Authorization_Procedure_and_Error_Code OMA-DM-LightweightM2M-2012-0104R01- CR_Registration_Interface
	30 Nov 2012		OMA-DM-LightweightM2M-2012-0107R01- CR_Appendix_for_LWM2M_Objects. OMA-DM-LightweightM2M-2012-0106R02- CR_Information_Interfaces. OMA-DM-LightweightM2M-2012-0108R01- CR_LWM2M_Server_Account_Object. OMA-DM-LightweightM2M-2012-0109R01- CR_Authorization_Update
	06 Dec 2012	6	OMA-DM-LightweightM2M-2012-0110R01- CR_Interfaces_Intro_Update
	19 Dec 2012	6,7,8,9, Annex	OMA-DM-LightweightM2M-2012-0111R01- CR_Object_Instance_Introduction OMA-DM-LightweightM2M-2012-0112-CR_Object_Template_Update OMA-DM-LightweightM2M-2012-0113R02-CR_Access_Control OMA-DM-LightweightM2M-2012-0114- CR_Update_Operation_Modification OMA-DM-LightweightM2M-2012-0115-CR_Connection_Control
	22 Jan 2013	2, 7, 8, 9, Annex	OMA-DM-LightweightM2M-2012-0101R03- CR_change_of_the_TLV_data_format OMA-DM-LightweightM2M-2012-0117- CR_remove_example_objects_and_resources OMA-DM-LightweightM2M-2013-0001R04-CR_Firmware_Object OMA-DM-LightweightM2M-2013-0003R01- CR_LwM2M_Client_and_Server_Security_Considerations OMA-DM-LightweightM2M-2013-0006- CR_Security_Mode_in_RessourceInfo_Table
	06 Feb 2013		OMA-DM-LightweightM2M-2013-0004R03-CR_SmartCard_Bootstrap OMA-DM-LightweightM2M-2013-0005R01-CR_device_object OMA-DM-LightweightM2M-2013-0007-CR_Object_Instance_Modification

Document Identifier	Date	Sections	Description
	26 Feb 2013	All	OMA-DM-LightweightM2M-2013-0002R04-CR_Adding_Creatable_Object OMA-DM-LightweightM2M-2013-0008R02-CR_Improvement_to_the_JSON_format_for_IETF_alignment OMA-DM-LightweightM2M-2013-0013R01-CR_LWM2M_Version_CoAP_Option OMA-DM-LightweightM2M-2013-0014R01-CR_Data_Format_Negotiation OMA-DM-LightweightM2M-2013-0015R02-CR_Notification_Aggregation_and_Reporting OMA-DM-LightweightM2M-2013-0016R03-CR_Connectivity OMA-DM-LightweightM2M-2013-0019R01-CR_SmartCard_Bootstrap_Appendix OMA-DM-LightweightM2M-2013-0020-CR_Response_Code
	01 Mar 2013	All	OMA-DM-LightweightM2M-2013-0011R03-CR_Failure_indication_for_firmware_object OMA-DM-LightweightM2M-2013-0022R03-CR_TLV_Tags OMA-DM-LightweightM2M-2013-0023R01-CR_location_object
	14 Mar 2013	All	OMA-DM-LightweightM2M-2012-0116R03-CR_Bootstrap_Interface_Chapter_Modification OMA-DM-LightweightM2M-2013-0018R01-CR_Bootstrap_Interface_Transport_Binding OMA-DM-LightweightM2M-2013-0024R04-CR_Time_Resource OMA-DM-LightweightM2M-2013-0026R05-CR_Erro_Code OMA-DM-LightweightM2M-2013-0027R01-CR_Delete_Object_Instance OMA-DM-LightweightM2M-2013-0028R01-CR_Location_objet_speed_direction
	09 Apr 2013	All	OMA-DM-LightweightM2M-2013-0047R02-CR_major_TS_cleanup OMA-DM-LightweightM2M-2013-0029R01-CR_Server_Object_Instance_Deletion OMA-DM-LightweightM2M-2013-0030R01-CR_Registration_Update OMA-DM-LightweightM2M-2013-0032-CR_Read_Operation_Update OMA-DM-LightweightM2M-2013-0044R01-CR_Response_Code_Update OMA-DM-LightweightM2M-2013-0034R01-CR_Device_Object_Update OMA-DM-LightweightM2M-2013-0035R01-CR_Bootstrap_Interface_Update OMA-DM-LightweightM2M-2013-0037R01-CR_Access_Control_Update OMA-DM-LightweightM2M-2013-0038R02-CR_Firmware_Object_Update OMA-DM-LightweightM2M-2013-0039-CR_Moving_Response_Code_Chapter
	12 Apr 2013	All	OMA-DM-LightweightM2M-2013-0054R02-CR_Bootstrap_Process_Update OMA-DM-LightweightM2M-2013-0051-CR_certificate_definition OMA-DM-LightweightM2M-2013-0052-CR_root_resource OMA-DM-LightweightM2M-2013-0053R01-CR_connectivity_object_update OMA-DM-LightweightM2M-2013-0050-CR_object_template_and_datatypes OMA-DM-LightweightM2M-2013-0036R02-CR_Information_Reporting_Update OMA-DM-LightweightM2M-2013-0049R01-CR_queue_mode

Document Identifier	Date	Sections	Description
	22 May 2013	All	OMA-DM-LightweightM2M-2013-0062R01-CR_TS_Editorial_streamlining OMA-DM-LightweightM2M-2013-0041R02-CR_Update_Error_Code OMA-DM-LightweightM2M-2013-0048R03-CR_Statistician_Object OMA-DM-LightweightM2M-2013-0055R01-CR_Cancel_Observation OMA-DM-LightweightM2M-2013-0056R01-CR_TLV_update OMA-DM-LightweightM2M-2013-0057-CR_SMS_trigger OMA-DM-LightweightM2M-2013-0058R06-CR_Security_key_formats OMA-DM-LightweightM2M-2013-0059-CR_Adding_Create_Operation_Example OMA-DM-LightweightM2M-2013-0061R01-CR_Adding_Access_Control_Example OMA-DM-LightweightM2M-2013-0063R01-CR_Reserved_Resource_ID_Space OMA-DM-LightweightM2M-2013-0064-CR_Update_Server_Deletion Plus editorial changes done by the editor
	10 Jun 2013	All	OMA-DM-LightweightM2M-2013-0070-CR_examples_update OMA-DM-LightweightM2M-2013-0066R01-CR_16bit_instance_IDs OMA-DM-LightweightM2M-2013-0067-CR_Observe_Operation_Range OMA-DM-LightweightM2M-2013-0068-CR_Server_Initiated_Bootstrap_Procedure OMA-DM-LightweightM2M-2013-0069R03-CR_observe_read_parameters OMA-DM-LightweightM2M-2013-0071R03-CR_Endpoint_Client_Name_–_Type_Attribute OMA-DM-LightweightM2M-2013-0072-CR_TS_X509_Validation_Rules OMA-DM-LightweightM2M-2013-0073-CR_Security_Section_Update OMA-DM-LightweightM2M-2013-0075-CR_Mandatory_Fields_and_Object_Template_Update OMA-DM-LightweightM2M-2013-0076R01-CR_Queue_Mode_Clarification OMA-DM-LightweightM2M-2013-0078R02-CR_BootstrapInformation___Objects OMA-DM-LightweightM2M-2013-0079R01-CR_Appendix_F_upgrade OMA-DM-LightweightM2M-2013-0081R02-CR_LWM2M_Security_Implications OMA-DM-LightweightM2M-2013-0082-CR_Data_Format_for_Resource_Supporting_Multiple_Instances OMA-DM-LightweightM2M-2013-0083R01-CR_no_sec_mode OMA-DM-LightweightM2M-2013-0084-CR_firmware_URI OMA-DM-LightweightM2M-2013-0085R01-CR_power_info_improvements OMA-DM-LightweightM2M-2013-0087R01-CR_Data_Type_Usage_Cleaning Plus editorial changes done by the editor
	17 Jul 2013	All	It incorporates: OMA-DM-LightweightM2M-2013-0103-CR_A103_TLV_bit_ordering OMA-DM-LightweightM2M-2013-0098-CR_A132 OMA-DM-LightweightM2M-2013-0097-CR_A046_A099_A100_A101_A104_A118_A119_139_140 OMA-DM-LightweightM2M-2013-0096R01-CR_A047_SC_Secure_Channel OMA-DM-LightweightM2M-2013-0095-CR_A180_Appendix_F OMA-DM-LightweightM2M-2013-0094R01-CR_Addressing_Comment_A187 OMA-DM-LightweightM2M-2013-0092R03-CR_Appendix_D_Fixes OMA-DM-LightweightM2M-2013-0089R01-CR_TLV_and_Device_Object_Examples_Fixes Plus editorial changes done by Seongyoon on behalf of the editor

Document Identifier	Date	Sections	Description
	02 Aug 2013	All	OMA-DM-LightweightM2M-2013-0100R02-CR_121 OMA-DM-LightweightM2M-2013-0101R01-CR_Client_and_Server_Initiated_Bootstrap_Update OMA-DM-LightweightM2M-2013-0102R01-CR_Resolving_Comments_on_Section_5.2 OMA-DM-LightweightM2M-2013-0108-CR_D.4_Clarification OMA-DM-LightweightM2M-2013-0110-CR_Secure_Channel_Fix Plus editorial changes done by the editor
	19 Aug 2013	All	OMA-DM-LightweightM2M-2013-0104R03-CR_Registration_Binding OMA-DM-LightweightM2M-2013-0106R02-CR_OMA_DM_LightweightM2M_2013_0102_CR_Resolving_Comments_on_Section_5.3_5.4 OMA-DM-LightweightM2M-2013-0111R03-CR_Resolving_Comments_on_Section_6 OMA-DM-LightweightM2M-2013-0112-CR_Server_Objects_Modifications OMA-DM-LightweightM2M-2013-0113-CR_D.2_Clarification OMA-DM-LightweightM2M-2013-0116-CR_TLV_Example_Editorial_Fix Plus editorial changes done by the editor
	28 Aug 2013	All	OMA-DM-LightweightM2M-2013-0099R03-CR_Resolving_Comments_on_Section_5.1 OMA-DM-LightweightM2M-2013-0109R05-CR_ACL_Clarification_Proposal_D3 OMA-DM-LightweightM2M-2013-0117R01-CR_security_comments_A114_A117_A120_A124 Plus editorial changes done by the editor
	04 Sep 2013	All	OMA-DM-LightweightM2M-2013-0114R01-CR_Comments_Resolving_for_D.3 OMA-DM-LightweightM2M-2013-0119R01-CR_Example_Client_Fix Plus editorial changes done by the editor
	12 Sep 2013	All	OMA-DM-LightweightM2M-2013-0124R02-CR_Reserved_ID OMA-DM-LightweightM2M-2013-0126-CR_registration_update_trigger_comment_A062 OMA-DM-LightweightM2M-2013-0122R01-CR_Firmware_Object_Fix
	17 Sep 2013		OMA-DM-LightweightM2M-2013-0127-CR_Connectivity_Monitoring_Object_comments_A172_A173 OMA-DM-LightweightM2M-2013-0128-CR_TLV_nesting_comment_A106 Plus editorial changes done by the editor

Document Identifier	Date	Sections	Description
	06 Oct 2013	1, 2, 4, 5, 5.1, 5.2, 5.2.1, 5.2.3, 5.3, 5.3.3-5.3.7, 5.4, 6.3.3.2, 6.3.4, 7.1, 7.1.3, 7.2, 7.2.1, 7.2.2, 7.2.2.1, 7.2.2.3, 7.2.2.4, 8.2, 8.2.2, 8.2.3, 8.2.4, 8.2.5, 8.3, 8.4, 8.5, B, C, D, D.1, D.2, D.2.1, E, E.1, E.1.2, E.2-E.8, F, G.1, G.2.2	Incorporated CRs: OMA-DM-LightweightM2M-2013-0123R03-CR_Cancel_Observation_Fix OMA-DM-LightweightM2M-2013-0129R03-CR_Introduction_Chapter_Update OMA-DM-LightweightM2M-2013-0130-CR_ACL_Term_Consistency OMA-DM-LightweightM2M-2013-0131-CR_Section_8.2.4_Update OMA-DM-LightweightM2M-2013-0132R01-CR_Queue_Mode_Chapter_Update OMA-DM-LightweightM2M-2013-0133-CR_SCR OMA-DM-LightweightM2M-2013-0135R01-CR_Observe_Clarification OMA-DM-LightweightM2M-2013-0137-CR_Response_Code_Update OMA-DM-LightweightM2M-2013-0138R01-CR_Object_Template_Update OMA-DM-LightweightM2M-2013-0141-CR_fixes_Appendix_B_C_D_E_F OMA-DM-LightweightM2M-2013-0142R04-CR_Access_Control_Object_Management OMA-DM-LightweightM2M-2013-0143-CR_Write_Attributes_comment_A079_A80_A81 OMA-DM-LightweightM2M-2013-0144R01-CR_Chap_1_and_2_A011_A012_A013 OMA-DM-LightweightM2M-2013-0145R02-CR_Figure_Update OMA-DM-LightweightM2M-2013-0146R02-CR_Execute_operation_arguments_A019 OMA-DM-LightweightM2M-2013-0147R01-CR_chap_6_4_2_A109_A110_A111 OMA-DM-LightweightM2M-2013-0149R01-CR_AI_Cancel_Observation OMA-DM-LightweightM2M-2013-0150-CR_Client_Registration_Term_Consistency OMA-DM-LightweightM2M-2013-0151R01-CR_UTC_offset_comment_A168 OMA-DM-LightweightM2M-2013-0153-CR_Appendix_F_Fix OMA-DM-LightweightM2M-2013-0155-CR_Missing_Normative_Texts OMA-DM-LightweightM2M-2013-0156-CR_Resource_Instances_A018_A073 OMA-DM-LightweightM2M-2013-0157R01-CR_Closing_LGE_Action_Items OMA-DM-LightweightM2M-2013-0158-CR_Partial_Update_Support OMA-DM-LightweightM2M-2013-0159-CR_Observe_Attribute_Clarification_A090_A147 OMA-DM-LightweightM2M-2013-0160R01-CR_Cancel_Observe_2_A085 Editorial changes
	17 Oct 2013	All	OMA-DM-LightweightM2M-2013-0154-CR_SCR_Table_UpdateSC Plus editorial changes done by the editor.

Document Identifier	Date	Sections	Description
	31 Oct 2013	3.2, 5, 5.1.2.3, 5.2.2, 5.3, 5.3.1, 5.3.3, 5.3.4, 5.4.2, 7.1.3, 7.1.4, 7.2, 7.2.1.1, 7.2.1.2, 7.2.2, 8.2.2, 8.2.4, 8.5, 8.6, B.1.2, B.1.6, B.1.7, B.2.2, B.2.3, B.2.7, D.1, E, F	Incorporated CRs: OMA-DM-LightweightM2M-2013-0162R03-CR_Bug_Fixes OMA-DM-LightweightM2M-2013-0163-CR_tool_generated_LWM2M_Objects OMA-DM-LightweightM2M-2013-0168-CR_Bug_Fix Editorial changes
	05 Nov 2013	7.2, E	Incorporated CRs: OMA-DM-LightweightM2M-2013-0139R04-CR_SMS_security_comments_A016_A112_A113
	03 Dec 2013	D	Incorporated CR: OMA-DM-LightweightM2M-2013-0171-CR_XML_schema_reference
Candidate Version OMA-TS-LightweightM2M-V1_0	10 Dec 2013	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2013-0368-INP_LightweightM2M_V1_0_ERP_and_ETR_for_Candidate_approval
Draft Versions OMA-TS-LightweightM2M-V1_0	09 Jan 2014	2.1, 5.3.4	Incorporated CR: OMA-DM-LightweightM2M-2014-0002R01-CR_Reference_To_IETF Editorial changes
	15 Jan 2014	5.3.2, 6.3.1, 6.3.3.1, 6.3.4	Incorporated CR: OMA-DM-LightweightM2M-2014-0001R02-CR_OIID_bugfix
	16 Apr 2014	5.1.4, 6.2, 6.3.1	Incorporated CR: OMA-DM-LightweightM2M-2014-0007R01-CR_Fix_Minor_Editorial_issues
	28 Apr 2014	E.4	Incorporated CR: OMA-DM-LightweightM2M-2014-0008R02-CR_FIX_DeviceObject__Section
	07 May 2014	E.4	Modify the editorial issues when incorporating CR0008R02
	13 May 2014	E.4	Incorporated CR: OMA-DM-LightweightM2M-2014-0009R02-CR_Fix_Device_Resource_Description
	19 Jun 2014	6.3.1, 6.3.3	Incorporated CR: OMA-DM-LightweightM2M-2014-0021-CR_update_appendix_references
	26 Aug 2014	E.6	Incorporated CR: OMA-DM-LightweightM2M-2014-0027R02-CR_Firmware_Update_Fix_for_coherency
	17 Sep 2014	6.3.4, 7.3.2, D.1, E, E.6	Incorporated CR: OMA-DM-LightweightM2M-2014-0039-CR_Few_fixes__FW_Appendix_ref
	01 Oct 2014	6.3.3.2, C	Incorporated CR: OMA-DM-LightweightM2M-2014-0042R03-CR_TLV_fix_and_Object_Link
	06 Oct 2014	6.3.3.3, 6.3.4, C	Incorporated CR: OMA-DM-LightweightM2M-2014-0043R02-CR_TLV___JSON_Illustration_w_Obj_Link_Res
	07 Oct 2014	6.3.4, C	Modify the mistakes when incorporating CR0043R02

Document Identifier	Date	Sections	Description
	11 Nov 2014	2, 4, 7.2, 7.2.2, 7.2.2.1, 7.2.2.2, 7.2.2.3, E.1	Incorporated CRs: OMA-DM-LightweightM2M-2014-0048R02-CR_E1_LWM2M_object_security_mandate OMA-DM-LightweightM2M-2014-0053R03-CR_SMS_DTLS_based_Security_on_Device OMA-DM-LightweightM2M-2014-0056R01-CR_add_missing_TR_069_reference Editorial changes
	26 Nov 2014	6.3.4	Incorporated CR: OMA-DM-LightweightM2M-2014-0060R01-CR_JSON_examples_Fix
Candidate Version OMA-TS-LightweightM2M-V1_0	26 Nov 2014	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2014-0271-INP_LightweightM2M_V1_0_ERP_for_Notification
Draft Versions OMA-TS-LightweightM2M-V1_0	02 Feb 2015	5.1, 5.1.2.3, 5.1.2.4, 5.1.3, 7.1, E.1	Incorporated CR: OMA-DM-LightweightM2M-2014-0066R03-CR_bootstrap_clarifications Editorial changes
	12 Feb 2015	5.3.4, 5.3.6, 5.4.3, 8.2.3, 8.2.5, 8.5	Incorporated CR: OMA-DM-LightweightM2M-2015-0004-CR_Various_Erratas
	25 Feb 2015	8.2.3, 8.2.4, 8.2.5, 8.3, 8.5	Incorporated CR: OMA-DM-LightweightM2M-2015-0007R01-CR_Erratas_Complement
	28 Feb 2015	5.1, 5.4.2, 5.4.4, 7.2.2.3, 8.2.4, D.1	Incorporated CRs: OMA-DM-LightweightM2M-2015-0008R01-CR_Attributes_ OMA-DM-LightweightM2M-2015-0011-CR_Objlnk_missing_in_object_template OMA-DM-LightweightM2M-2015-0012-CR_LWM2M_UICC_Application_Tar_Value
	17 Mar 2015	6.3, 6.3.1, 6.3.2, 6.3.4, 8.2.4	Incorporated CR: OMA-DM-LightweightM2M-2015-0009R02-CR_Content_Type Editorial changes
	18 Mar 2015	E.4	Incorporated CR: OMA-DM-LightweightM2M-2015-0010R02-CR_Device_Info_Fix
	22 Apr 2015	E.1, E.1.1, E.1.1.1	Incorporated CRs: OMA-DM-LightweightM2M-2015-0017R01-CR_CipherSuites Corrected incorporation of CRs: OMA-DM-LightweightM2M-2015-0007R01-CR_Erratas_Complement OMA-DM-LightweightM2M-2015-0008R01-CR_Attributes_ Editorial changes
	13 May 2015	5.3.1, E.4	Incorporated CRs: OMA-DM-LightweightM2M-2015-0016R01-CR_Host_Device_Resource_Parameters OMA-DM-LightweightM2M-2015-0020R01-CR_Registration_Information Editorial changes

Document Identifier	Date	Sections	Description
	15 Jun 2015	3.2, 5.1, 5.2.1, 5.2.2.3, 5.2.2.4, 5.4.6, 6.3.4, 7.3.2, 7.3.2.1, 7.3.2.3, 7.3.2.4, 8.2.2, 8.2.4, 8.2.5, 8.5, E.2	Incorporated CRs: OMA-DM-LightweightM2M-2015-0025R02-CR_Bootstrap_Finish OMA-DM-LightweightM2M-2015-0026R01-CR_Generic_Error_Code OMA-DM-LightweightM2M-2015-0027-CR_Default_Minimum_Period OMA-DM-LightweightM2M-2015-0028R01-CR_Access_Rights___Authorization_Consistency OMA-DM-LightweightM2M-2015-0029-CR_JSON OMA-DM-LightweightM2M-2015-0031R01-CR_Queue_Mode OMA-DM-LightweightM2M-2015-0033R03-CR_R_Only_Resource_Clarification OMA-DM-LightweightM2M-2015-0034R03-CR_bootstrap_clarifications OMA-DM-LightweightM2M-2015-0035R02-CR_Attribute_TypeFixes Editorial changes
	07 Jul 2015	7.1.3	Incorporated CR: OMA-DM-LightweightM2M-2015-0040R02-CR_x509_editorial
	03 Sep 2015	5, 5.2.2.3, 5.2.2.4, 5.2.5, 5.4.3, 5.4.5, 6.3.3, 6.3.4, 8.2.2, 8.2.4, C, D.1, E.6, I	Incorporated CRs: OMA-DM-LightweightM2M-2015-0043R01-CR_Firmware_Update_Fix_ OMA-DM-LightweightM2M-2015-0044R04-CR_Exec_Res__Arg_Spec OMA-DM-LightweightM2M-2015-0046R02-CR_BootstrapIntf OMA-DM-LightweightM2M-2015-0047R02-CR_TLV_and_JSON_Usages_fixes OMA-DM-LightweightM2M-2015-0048R01-CR_TLV___JSON_Madia_Type_IANA_request OMA-DM-LightweightM2M-2015-0050-CR_TLV_Signed_Integer Editorial changes
	10 Sep 2015	5.4, 5.4.6, 7.3.1.2.1	Incorporated CR: OMA-DM-LightweightM2M-2015-0054R02-CR_Commands_Definition_Conflicts
Candidate Version OMA-TS-LightweightM2M-V1_0	30 Oct 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0177-INP_LightweightM2M_V1_0_ERP_for_Notification
Draft Versions OMA-TS-LightweightM2M-V1_0	16 Nov 2015	8.4, I	Incorporated CRs: OMA-DM-LightweightM2M-2015-0056R01-CR_sec84_mismatch_picture OMA-DM-LightweightM2M-2015-0057R01-CR_LWM2M_Media_Types_Registration_Fixes Editorial changes
	01 Dec 2015	3.3, 7.1.1, 7.3.1.2, 8.3, E.2, E.3, E.5, E.7, E.8, all	Incorporated CRs: OMA-DM-LightweightM2M-2015-0059-CR_queue_mode_clarification OMA-DM-LightweightM2M-2015-0061-CR_editorials_on_20151116 OMA-DM-LightweightM2M-2015-0062-CR_support_info_for_TLS_PSK_SHA256 OMA-DM-LightweightM2M-2015-0063-CR_Conn_stats_coll_duration Removed unnecessary spaces shown by default as grammatical errors by MS word.
Candidate Version OMA-TS-LightweightM2M-V1_0	01 Dec 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0206-INP_LightweightM2M_V1_0_ERP_for_Notification

Document Identifier	Date	Sections	Description
Draft Version OMA-TS-LightweightM2M-V1_0	11 Dec 2015	5.4.6, 7.3, 7.3.1.1, 7.3.1.2.1, 7.3.2, 7.3.2.1, 7.3.2.2, 7.3.2.3, 7.3.2.4	Incorporated CR: OMA-DM-LightweightM2M-2015-0065R01-CR_MultiSrv_Context_Security
Candidate Version OMA-TS-LightweightM2M-V1_0	14 Dec 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0221-INP_LightweightM2M_V1_0_ERP_for_Notification
Draft Versions OMA-TS-LightweightM2M-V1_0	25 Jan 2016	5.1.2, 8.2.4, C, D.1, E.1, E.5, E.8	Incorporated CRs: OMA-DM-LightweightM2M-2015-0068-CR_Object_Template_Fix OMA-DM-LightweightM2M-2015-0069-CR_Security_Object_Fix OMA-DM-LightweightM2M-2016-0001-CR_Notification_Attributes_Reset OMA-DM-LightweightM2M-2016-0003-CR_TLV_float OMA-DM-LightweightM2M-2016-0004-CR_Conn_monitor OMA-DM-LightweightM2M-2016-0005R01-CR_Conn_statistics OMA-DM-LightweightM2M-2016-0010-CR_default_pmin
	02 Feb 2016	5.4.3, 8.2.4	Incorporated CR: OMA-DM-LightweightM2M-2016-0007-CR_Partial_Update
	08 Mar 2016	6.3, 6.3.3, 6.3.4, 8.1, 8.5	Incorporated CR: OMA-DM-LightweightM2M-2016-0012R04-CR_Clarify_content_format_and_make_TLV_default
	07 Apr 2016	5.2.5, 5.2.5.1, 5.2.5.3, 5.2.5.4, 5.3, 5.4, 5.5, 8.2.2, 8.2.3, 8.2.4, 8.2.5, 8.5	Incorporated CRs: OMA-DM-LightweightM2M-2016-0015R03-CR_Response_Codes_Fixes OMA-DM-LightweightM2M-2016-0025R01-CR_Fix_Bootstrap_Commands OMA-DM-LightweightM2M-2016-0026-CR_Fix_Chap_5 OMA-DM-LightweightM2M-2016-0027-CR_Fix_Chap_8
Candidate Version OMA-TS-LightweightM2M-V1_0	07 Apr 2016	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2016-0056-INP_LightweightM2M_V1_0_ERP_for_Notification

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for LWM2M Client

B.1.1 Bootstrap Interface

Item	Function	Reference	Requirement
LWM2M-BOOT-001-C-M	Support of at least one Bootstrap Mode	Section 5.1	
LWM2M-BOOT-002-C-O	Support of Factory Bootstrap Mode	Section 5.2.2.1	
LWM2M-BOOT-003-C-O	Support of Bootstrap from Smartcard	Section 5.2.2.2, Appendix F	LWM2M-BOOT-012C-O
LWM2M-BOOT-004-C-O	Support of Client Initiated Bootstrap	Section 5.2.2.3	
LWM2M-BOOT-005-C-O	Support of Server Initiated Bootstrap	Section 5.2.2.4	
LWM2M-BOOT-006-C-M	Support of LWM2M Server Bootstrap Information	Section 5.2.1	
LWM2M-BOOT-007-C-O	Support of LWM2M Bootstrap Server Bootstrap Information	Section 5.2.1	
LWM2M-BOOT-008-C-M	Support of accepting Bootstrap Information transferred	Section 5.2.1	
LWM2M-BOOT-009-C-M	Support of Bootstrap Sequence	Section 5.2.3	
LWM2M-BOOT-010-C-M	Support of Bootstrap Security	Section 5.2.4	
LWM2M-BOOT-011-C-O	Support of Bootstrap from Smartcard with Secure Channel	Section 5.2.2.2, Appendix F	LWM2M-BOOT-012C-O AND LWM2M-SEC-007-C-O
LWM2M-BOOT-012-C-O	Retrieve & Process bootstrap data from Smartcard	Section 5.2.2.2	
LWM2M-BOOT-013-C-O	Check for Bootstrap Data change in Smartcard	Section 5.2.2.2	

B.1.2 Client Registration

Item	Function	Reference	Requirement
LWM2M-CR-001-C-M	Support of “Register” operation	Section 5.3.1	
LWM2M-CR-002-C-M	Support of Endpoint Client Name parameter	Section 5.3.1	
LWM2M-CR-003-C-M	Support of Lifetime parameter	Section 5.3.1	
LWM2M-CR-004-C-O	Support of LWM2M Version parameter	Section 5.3.1	

Item	Function	Reference	Requirement
LWM2M-CR-005-C-M	Support of Binding Mode parameter	Section 5.3.1, 5.3.1.1	
LWM2M-CR-006-C-O	Support of SMS Number parameter	Section 5.3.1	
LWM2M-CR-007-C-M	Support of Object and Object Instances parameter	Section 5.3.1	
LWM2M-CR-008-C-M	Support of “Update” operation	Section 5.3.2	
LWM2M-CR-009-C-O	Support of “De-register” operation	Section 5.3.3	
LWM2M-CR-010-C-O	Support of Updating Bootstrap Information from Smartcard at Register/Update	Section 5.2.2.2	(LWM2M-CR-001-C-M OR LWM2M-CR-008-C-M) AND LWM2M-BOOT-013-C-O AND (LWM2M-BOOT-003-C-O OR LWM2M-BOOT-011-C-O)

B.1.3 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LWM2M-DMSE-001-C-M	Support of “Read” operation	Section 5.4.1	
LWM2M-DMSE-002-C-M	Support of “Discover” operation	Section 5.4.2	
LWM2M-DMSE-003-C-M	Support of “Write” operation	Section 5.4.2	
LWM2M-DMSE-004-C-M	Support of “Write Attributes” operation	Section 5.4.4	
LWM2M-DMSE-005-C-O	Support of Minimum Period parameter	Section 5.4.4	
LWM2M-DMSE-006-C-O	Support of Maximum Period parameter	Section 5.4.4	
LWM2M-DMSE-007-C-O	Support of Greater Than parameter	Section 5.4.4	
LWM2M-DMSE-008-C-O	Support of Less Than parameter	Section 5.4.4	
LWM2M-DMSE-009-C-O	Support of Step parameter	Section 5.4.4	
LWM2M-DMSE-010-C-O	Support of Cancel parameter	Section 5.4.4	
LWM2M-DMSE-011-C-M	Support of “Execute” operation	Section 5.4.5	
LWM2M-DMSE-012-C-M	Support of “Create” operation	Section 5.4.6	
LWM2M-DMSE-013-C-M	Support of “Delete” operation	Section 5.4.7	

B.1.4 Information Reporting

Item	Function	Reference	Requirement
LWM2M-IR-001-C-M	Support of "Observe" operation	Section 5.5.1	
LWM2M-IR-002-C-M	Support of "Notify" operation	Section 5.5.2	
LWM2M-IR-003-C-M	Support of "Cancel Observation" operation	Section 5.5.3	

B.1.5 Data Format

Item	Function	Reference	Requirement
LWM2M-DF-001-C-M	Support of Plain Text format	Section 6.3, 6.3.1	
LWM2M-DF-002-C-M	Support of Opaque format	Section 6.3, 6.3.2	
LWM2M-DF-003-C-M	Support of TLV format	Section 6.3, 6.3.3	
LWM2M-DF-004-C-O	Support of JSON format	Section 6.3, 6.3.4	

B.1.6 Security

Item	Function	Reference	Requirement
LWM2M-SEC-001-C-M	Support of at least one key mode	Section 7.1	LWM2M-SEC-002-C-O OR LWM2M-SEC-003-C-O OR LWM2M-SEC-004-C-O OR LWM2M-SEC-004-C-O
LWM2M-SEC-002-C-O	Support of Pre-Shared Keys mode	Section 7.1.1	
LWM2M-SEC-003-C-O	Support of Raw Public Key Certificates mode	Section 7.1.2	
LWM2M-SEC-004-C-O	Support of X.509 Certificates mode	Section 7.1.3	
LWM2M-SEC-005-C-O	Support of No Sec mode	Section 7.1.4	
LWM2M-SEC-006-C-O	Support of UDP Channel Security	Section 7.1	
LWM2M-SEC-007-C-O	Support of Smartcard Secure Channel	Section 7.1, Appendix G	LWM2M-SEC-009-C-O
LWM2M-SEC-008-C-O	Support of Access Control Mechanism	Section 7.3	
LWM2M-SEC-009-C-O	Smartcard Secure Channel using [GLOBALPLATFORM] [GP SCP03]		

B.1.7 Mechanism

Item	Function	Reference	Requirement
LWM2M-MEC-001-C-O	Support of Queue Mode	Section 8.3	
LWM2M-MEC-002-C-M	Support of UDP Binding	Section 8.6.1	
LWM2M-MEC-003-C-O	Support of SMS Binding	Section 8.6.2	

B.1.8 Objects

Item	Function	Reference	Requirement
LWM2M-OBJ-001-C-M	Support of LWM2M Security Object	Appendix E.1	
LWM2M-OBJ-002-C-M	Support of LWM2M Server Object	Appendix E.2	
LWM2M-OBJ-003-C-O	Support of Access Control Object	Appendix E.3	
LWM2M-OBJ-004-C-M	Support of Device Object	Appendix E.4	
LWM2M-OBJ-005-C-O	Support of Connectivity Monitoring Object	Appendix E.5	
LWM2M-OBJ-006-C-O	Support of Firmware Update Object	Appendix E.6	
LWM2M-OBJ-007-C-O	Support of Location Object	Appendix E.7	
LWM2M-OBJ-008-C-O	Support of Connectivity Statistics Object	Appendix E.8	

B.2 SCR for LWM2M Server

B.2.1 Bootstrap Interface

Item	Function	Reference	Requirement
LWM2M-BOOT-005-S-M	Support of Server Initiated Bootstrap	Section 5.2.2.4	
LWM2M-BOOT-010-S-M	Support of Bootstrap Security	Section 5.2.4	

B.2.2 Client Registration

Item	Function	Reference	Requirement
LWM2M-CR-001-S-M	Support of “Register” operation	Section 5.3.1	
LWM2M-CR-002-S-M	Support of Endpoint Client Name parameter	Section 5.3.1	
LWM2M-CR-003-S-M	Support of Lifetime parameter	Section 5.3.1	
LWM2M-CR-004-S-M	Support of LWM2M Version parameter	Section 5.3.1	
LWM2M-CR-005-S-M	Support of Binding Mode parameter	Section 5.3.1, 5.3.1.1	
LWM2M-CR-006-S-M	Support of SMS Number parameter	Section 5.3.1	
LWM2M-CR-007-S-M	Support of Object and Object Instances parameter	Section 5.3.1	
LWM2M-CR-001-S-M	Support of “Update” operation	Section 5.3.2	
LWM2M-CR-001-S-M	Support of “De-register” operation	Section 5.3.3	

B.2.3 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LWM2M-DMSE-001-S-M	Support of “Read” operation	Section 5.4.1	
LWM2M-DMSE-002-S-M	Support of “Discover” operation	Section 5.4.2	
LWM2M-DMSE-003-S-M	Support of “Write” operation	Section 5.4.2	
LWM2M-DMSE-004-S-M	Support of “Write Attributes” operation	Section 5.4.4	
LWM2M-DMSE-005-S-M	Support of Minimum Period parameter	Section 5.4.4	
LWM2M-DMSE-006-S-M	Support of Maximum Period parameter	Section 5.4.4	
LWM2M-DMSE-007-S-M	Support of Greater Than parameter	Section 5.4.4	
LWM2M-DMSE-008-S-M	Support of Less Than parameter	Section 5.4.4	
LWM2M-DMSE-009-S-M	Support of Step parameter	Section 5.4.4	
LWM2M-DMSE-010-S-M	Support of “Execute” operation	Section 5.4.5	
LWM2M-DMSE-011-S-M	Support of “Create” operation	Section 5.4.6	
LWM2M-DMSE-012-S-M	Support of “Delete” operation	Section 5.4.7	

B.2.4 Information Reporting

Item	Function	Reference	Requirement
LWM2M-IR-001-S-M	Support of “Observe” operation	Section 5.5.1	
LWM2M-IR-002-S-M	Support of “Notify” operation	Section 5.5.2	
LWM2M-IR-003-S-M	Support of “Cancel Observation” operation	Section 5.5.3	

B.2.5 Data Format

Item	Function	Reference	Requirement
LWM2M-DF-001-S-M	Support of Plain Text format	Section 6.3, 6.3.1	
LWM2M-DF-002-S-M	Support of Opaque format	Section 6.3, 6.3.2	
LWM2M-DF-003-S-M	Support of TLV format	Section 6.3, 6.3.3	
LWM2M-DF-004-S-M	Support of JSON format	Section 6.3, 6.3.4	

B.2.6 Security

Item	Function	Reference	Requirement
LWM2M-SEC-002-S-M	Support of Pre-Shared Keys mode	Section 7.1.1	
LWM2M-SEC-003-S-M	Support of Raw Public Key Certificates mode	Section 7.1.2	
LWM2M-SEC-004-S-M	Support of X.509 Certificates mode	Section 7.1.3	
LWM2M-SEC-005-S-M	Support of No Sec mode	Section 7.1.4	
LWM2M-SEC-006-S-M	Support of UDP Channel Security	Section 7.1	

B.2.7 Mechanism

Item	Function	Reference	Requirement
LWM2M-MEC-001-S-M	Support of Queue Mode	Section 8.3	
LWM2M-MEC-002-S-M	Support of UDP Binding	Section 8.6.1	
LWM2M-MEC-003-S-O	Support of SMS Binding	Section 8.6.2	

B.2.8 Objects

Item	Function	Reference	Requirement
LWM2M-OBJ-001-S-M	Support of LWM2M Security Object	Appendix E.1	
LWM2M-OBJ-002-S-M	Support of LWM2M Server Object	Appendix E.2	
LWM2M-OBJ-003-S-O	Support of Access Control Object	Appendix E.3	
LWM2M-OBJ-004-S-M	Support of Device Object	Appendix E.4	
LWM2M-OBJ-005-S-O	Support of Connectivity Monitoring Object	Appendix E.5	
LWM2M-OBJ-006-S-O	Support of Firmware Update Object	Appendix E.6	
LWM2M-OBJ-007-S-O	Support of Location Object	Appendix E.7	
LWM2M-OBJ-008-S-O	Support of Connectivity Statistics Object	Appendix E.8	

Appendix C. Data Types (Normative)

This appendix defines the data types that a Resource can be defined to be.

Data Type	Description	Text Format	TLV Format
String	A UTF-8 string, the minimum and/or maximum length of the String MAY be defined.	Represented as a UTF-8 string.	Represented as a UTF-8 string of Length bytes.
Integer	An 8, 16, 32 or 64-bit signed integer. The valid range of the value for a Resource SHOULD be defined. This data type is also used for the purpose of enumeration.	Represented as an ASCII signed integer.	Represented as a binary signed integer in network byte order, and in two's complement representation. The value may be 1 (8-bit), 2 (16-bit), 4 (32-bit) or 8 (64-bit) bytes long as indicated by the Length field. When transmitted over network, the data is represented in network byte order (big endian).
Float	A 32 or 64-bit floating point value. The valid range of the value for a Resource SHOULD be defined.	Represented as an ASCII signed decimal.	Represented as an [IEEE 754-2008] [FLOAT] binary floating point value. The value may use the binary32 (4 byte Length) or binary64 (8 byte Length) format as indicated by the Length field. When transmitted over network, the data is represented in network byte order (big endian).
Boolean	An integer with the value 0 for False and the value 1 for True.	Represented as the ASCII value 0 or 1.	Represented as an Integer with value 0, or 1. The Length of a Boolean value MUST always be 1.
Opaque	A sequence of binary octets, the minimum and/or maximum length of the String MAY be defined.		Represented as a sequence of binary data of Length bytes.
Time	Unix Time. A signed integer representing the number of seconds since Jan 1st, 1970 in the UTC time zone.	Represented as an ASCII integer.	Same representation as Integer.
Objlnk	Object Link. The object link is used to refer an Instance of a given Object. An Object link value is composed of two concatenated 16-bits unsigned integers following the Network Byte Order convention. The Most Significant Halfword is an ObjectID, the Least Significant Hafword is an ObjectInstance ID. An Object Link referencing no Object Instance will contain the concatenation of 2 MAX-ID values (null link)	Represented as a UTF-8 string containing 2 16-bits ASCII integers separated by a ':' ASCII character.	Same representation as 2 16-bits Integer one beside the other. The first one represents the ObjectID, and the second one represents the ObjectInstanceID. This value is always 4 bytes long.
none	no specific data type affected to that resource: it exclusively concerns Executable Resource		

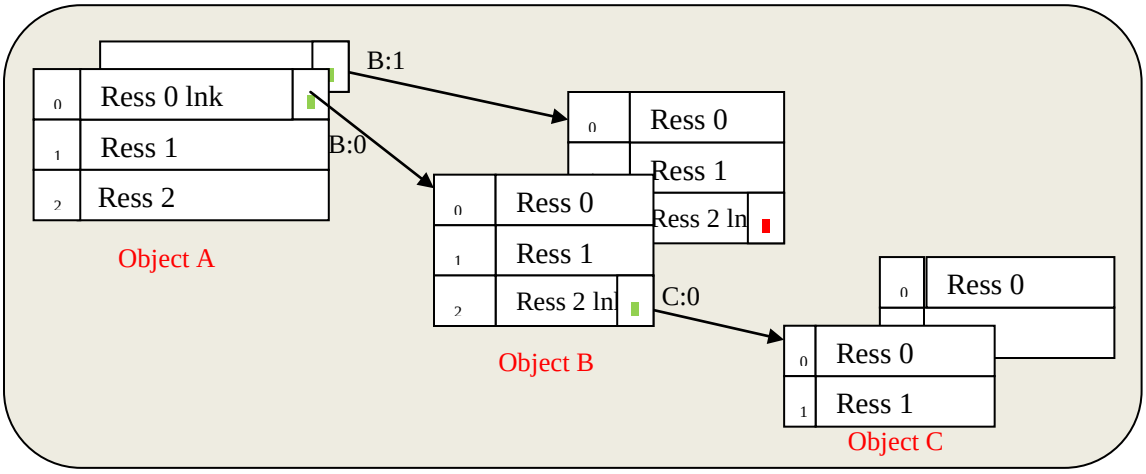


Figure 26: Object link Resource simple illustration

Appendix D. LWM2M Object Template and Guidelines (Normative)

This Appendix provides the template to be used for the specification of LWM2M Objects. Furthermore, guidelines for the creation of LWM2M Objects are provided.

The XML versions of LWM2M Objects MUST comply with the XML schema which can be found here: http://openmobilealliance.org/tech/profiles/LWM2M.xsd

D.1 Object Template

Appendix D.x LWM2M Object: *<LWM2M object name>*

Description

Object definition:

Name	Object ID	Instances	Mandatory	Object URN
Object Name	16-bit Unsigned Integer	Multiple/Single	Mandatory/Optional	urn:oma:lwm2m:{oma,ext,x}:{Object ID}

- **Name:** specifies the Object name.
- **Object ID:** specifies the Object ID.
- **Instances:** indicates whether this Object supports multiple Object Instances or not. If this field is "Multiple" then the number of Object Instance can be from 0 to many. If this field is "Single" then the number of Object Instance can be from 0 to 1. If the Object field "Mandatory" is "Mandatory" and the Object field "Instances" is "Single" then, the number of Object Instance MUST be 1.
- **Mandatory:** if this field is "Mandatory", then the LWM2M Client MUST support this Object. If this field is "Optional", then the LWM2M Client SHOULD support this Object.
- **Object URN:** specifies the Object URN. The format of the Object URN is "urn:oma:lwm2m:{oma,ext,x}:{Object ID}" and {} part means that those values are variable and filled with real value. For example, Object URN of LWM2M Server Object is "urn:oma:lwm2m:oma:1".

Resource definition:

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Resource Name	R (Read), W (Write), E (Execute)	Multiple/ Single	Mandatory /Optional	String, Integer, Float, Boolean, Opaque, Time, Objlnk none	If any	If any	Description

- **ID:** specifies the Resource ID which is unique within Object.
- **Name:** specifies the Resource name.
- **Operations:** indicates which operations the Resource supports in the "Device Management & Service Enablement" Interface. This field can be set to a combination of R (Read, Observe, Discover, Write Attributes), and W (Write), or can be set to E (Execute); Executable Operation is exclusive regarding the two others (R,W). This field may also have an empty value, which means that this field is not allowed to be accessed via "Device Management & Service Enablement" Interface but allowed to be accessed via "Bootstrap" Interface.

- **Instances:** indicates whether this Resource supports multiple Resource Instances or not. If this field is "Multiple" then the number of Resource Instance can be from 0 to many. If this field is "Single" then the number of Resource Instance can be from 0 to 1. If the Resource field "Mandatory" is "Mandatory" and the field "Instances" of the Resource is "Single" then, the number of Resource Instance MUST be 1. Resource which supports "Execute" operation MUST have "Single" as value of the "Instances" field.
- **Mandatory:** if this field is "Mandatory", then the LWM2M Server and the LWM2M Client MUST support the Resource. If this field is "Optional", then the LWM2M Server and the LWM2M Client SHOULD support the Resource.
- **Type:** Data Type indicates the type of Resource value. Data Types used in this enabler are described in Appendix C Data Types. Resource which supports "Execute" operation MUST have no associated Data Type (none)
- **Range or Enumeration:** this field limits the value of Resource.
- **Units:** specifies the unit of the Resource value.
- **Description:** specifies the Resource description.

In addition to the object and resource definition tables, an object containing Executable Resource(s) is specified in third Table, gathering the definition of the arguments of all the Executable Resources of that Object.

This table provides the properties of arguments

Executable Resource Arguments Definition

ID	Resource Name	Order	Name	Type	Range or Enum	Unit	Description
		[0:9]	String	LWM2M Data Types	If any	If any	

Example of an Executable Resource Arguments Definition Table for an Object having 3 Executable Resource

Execution Resource Arguments definitions

ID	Resource Name	Order	Name	Type	Range or Enum	Unit	Description
5	Delete	0	-	none	-	-	1 argument EXECUTE /X/0/5 0
7	Update	0	Remove	none	-		2 arguments Ex EXECUTE /X/0/7 0,1='2'
		1	Keep	Integer	[0-2]		
10	Create						

D.2 Open Mobile Naming Authority (OMNA) Guidelines

This appendix defines guidelines for OMNA regarding registries and protocol ID ranges to be maintained.

D.2.1 Object Registry

LWM2M Objects must be registered with the OMNA Lightweight Object registry. There are three classes of Objects in which an Object can be registered:

- OMA Objects (oma label) – Objects defined by the Open Mobile Alliance.
- 3rd Party Standards Development Organisation (SDO) Objects (ext label) – Objects defined by a 3rd party SDO.

- Vendor Specific Objects (x label) – Objects defined by a vendor or individual, such an Object may be either private (no DDF or Specification made available) or public.

Each one of these classes is assigned a range of IDs by OMNA.

The URN format for an Object is automatically built from the class of Object and the Object ID as follows:

urn:oma:lwm2m:{oma,ext,x}:{Object ID}

D.2.2 Resource Registry

LWM2M Objects are specified as being composed of Resources, each identified by a Resource ID. Resources can either be specific to each Object with meaning only when used in that Object, or Reusable Resources can be registered, assigned an ID from the OMNA range and re-used in any Object. The following Resource ID ranges are defined:

- Object specific Resource ID range – Defined by the Object specification.
- Reusable Resource ID range – Registered by an Object Specification, with the Resource ID assigned by OMNA. Defined in any Object specification. Resources from this Resource ID range can be re-used in any Object.
- Reserved range – Range or Resource IDs reserved for future use.

A Reusable Resource ID registration entry MUST define the Resource Name, Resource ID (assigned by OMNA), Supported Operations, Data Type, Range or Enumeration, Units and Description of the Resource.

Appendix E. LWM2M Objects defined by OMA (Normative)

This Appendix provides LWM2M Objects defined by OMA. Other organizations and companies may define additional LWM2M according to the guidelines and template provided in Appendix D.

The following LWM2M Objects have been defined by OMA as part of LWM2M 1.0:

Object	Object ID
LWM2M Security	0
LWM2M Server	1
Access Control	2
Device	3
Connectivity Monitoring	4
Firmware	5
Location	6
Connectivity Statistics	7

Table 26: LWM2M Objects defined by OMA LWM2M 1.0

The LWM2M Server MUST support LWM2M Security, LWM2M Server, and Device Object and SHOULD support Access Control, Device, Connectivity, Firmware Update, Location, and Connectivity Statistics Object.

E.1 LWM2M Object: LWM2M Security

Description

This LWM2M Object provides the keying material of a LWM2M Client appropriate to access a specified LWM2M Server. One Object Instance SHOULD address a LWM2M Bootstrap Server.
These LWM2M Object Resources MUST only be changed by a LWM2M Bootstrap Server or Bootstrap from Smartcard and MUST NOT be accessible by any other LWM2M Server.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
LWM2M Security	0	Multiple	Mandatory	urn:oma:lwm2m:oma:0

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	LWM2M Server URI		Single	Mandatory	String	0-255 bytes		Uniquely identifies the LWM2M Server or LWM2M Bootstrap Server, and is in the form: "coaps: //host:port", where host is an IP address or FQDN, and port is the UDP port of the Server.
1	Bootstrap Server		Single	Mandatory	Boolean			Determines if the current instance concerns a LWM2M Bootstrap Server (true) or a standard LWM2M Server (false)
2	Security Mode		Single	Mandatory	Integer	0-3		Determines which UDP payload security mode is used 0: Pre-Shared Key mode 1: Raw Public Key mode 2: Certificate mode 3: NoSec mode
3	Public Key or Identity		Single	Mandatory	Opaque			Stores the LWM2M Client's Certificate (Certificate mode), public key (RPK mode) or PSK Identity (PSK mode). The format is defined in Section E.1.1.
4	Server Public Key		Single	Mandatory	Opaque			Stores the LWM2M Server's or LWM2M Bootstrap Server's Certificate (Certificate mode), public key (RPK mode). The format is defined in Section E.1.1.
5	Secret Key		Single	Mandatory	Opaque			Stores the secret key or private key of the security mode. The format of the keying material is defined by the security mode in Section E.1.1. This Resource MUST only be changed by a bootstrap server and MUST NOT be readable by any server.
6	SMS Security Mode		Single	Optional	Integer	0-255		Determines which SMS security mode is used (see section 7.2) 0: Reserved for future use 1: DTLS mode (Device terminated) PSK mode assumed 2: Secure Packet Structure mode (S martcard terminated) 3: NoSec mode 4: Reserved mode (DTLS mode with multiplexing Security Association support) 5-203 : Reserved for future use 204-255: Proprietary modes
7	SMS Binding Key Parameters		Single	Optional	Opaque	6 bytes		Stores the KIc, KID, SPI and TAR. The format is defined in Section E.1.2.

8	SMS Binding Secret Key(s)		Single	Optional	Opaque	16-32-48 bytes		Stores the values of the key(s) for the SMS binding. This resource MUST only be changed by a bootstrap server and MUST NOT be readable by any server.
9	LWM2M Server SMS Number		Single	Optional	String			MSISDN used by the LWM2M Client to send messages to the LWM2M Server via the SMS binding. The LWM2M Client SHALL silently ignore any SMS originated from unknown MSISDN
10	Short Server ID		Single	Optional	Integer	1-65535		This identifier uniquely identifies each LWM2M Server configured for the LWM2M Client. This Resource MUST be set when the Bootstrap Server Resource has false value. Default Short Server ID (i.e. 0) MUST NOT be used for identifying the LWM2M Server.
11	Client Hold Off Time		Single	Optional	Integer		s	Relevant information for a Bootstrap Server only. The number of seconds to wait before initiating a Client Initiated Bootstrap once the LWM2M Client has determined it should initiate this bootstrap mode In case client initiated bootstrap is supported by the LWM2M Client, this resource MUST be supported.
12	Bootstrap Server Account Timeout		Single	Optional	Integer		s	The LWM2M Server SHOULD purge the LWM2M Bootstrap Server Account after successful bootstrapping. If it has not already been purged, the LWM2M Client MUST purge the LWM2M Bootstrap Server Account after the timeout value given by this resource. The lowest timeout value is 1. If the value is set to 0 the Bootstrap Server Account lifetime is infinite.

E.1.1 UDP Channel Security: Security Key Resource Format

This section defines the format of the Secret Key and Public Key and Identity Resources of the LWM2M Server and LWM2M Bootstrap Server Objects when using UDP Channel security. These Resources are used to configure the security mode and keying material that a Client uses with a particular Server. The Objects are configured on the Client using one of the Bootstrap mechanisms described in Section 5.1. The use of this keying material for each security mode is defined in Section 7.1.

E.1.1.1 Pre-Shared Key (PSK) Mode

The PSK is a binary shared secret key between the Client and Server of the appropriate length for the Cipher Suite used [RFC4279]. This key is composed of a sequence of binary bytes in the Secret Key Resource. The default PSK Cipher Suites defined in this specification use a 128-bit AES key. Thus this key would be represented in 16 bytes in the Secret Key Resource.
The corresponding PSK Identity for this PSK is stored in the Public Key or Identity Resource. The PSK Identity is simply stored as a UTF-8 String as per [RFC4279]. Clients and Servers MUST support arbitrary PSK Identities of up to 128 bytes and PSK keys of up to 64 bytes in length as required by [RFC4279].

E.1.1.2 Raw-Public Key (RPK) Mode

The raw-public key mode requires a public key and a private key of the appropriate type and length for the Cipher Suite used. These keys are carried as a sequence of binary bytes with the public key stored in the Public Key or Identity Resource, and the private key stored in the Secret Key Resource. The default RPK Cipher Suites defines in this specification use a 256-bit ECC key. Thus the Certificate Resource would contain a 32 byte public key and the Secret Key Resource a 32 byte private key.

E.1.1.3 Certificate Mode

The Certificate mode requires an X.509v3 Certificate along with a matching private key. The private key is stored in the Secret Key Resource as in RPK mode. The Certificate is simply represented as binary X.509v3 in the value of the Public Key or Identity Resource.

E.1.2 SMS Payload Security: Security Key Resource Format

This section defines the format of the Secret Key and Public Key and Identity resources of the LWM2M Server and LWM2M Bootstrap Objects when using SMS Payload security. These resources are used to configure keying material that a Client uses with a particular Server. The Objects are configured on the Client using one of the Bootstrap mechanisms described in Section 5.1. The use of this keying material is defined in Section 7.2.

The SMS key parameters are stored in the order KIc, KID, SPI, TAR (KIc is byte 0).

Ordering of bits within bytes SHALL follow ETSI TS 102 221, section 3.4 “Coding Conventions” (b8 MSB, b1 LSB).

E.1.3 Unbootstrapping

Unbootstrapping is the process of deleting a Security Object Instance. If a Security Object Instance is to be deleted, certain related resources and configurations need to be deleted or modified. Therefore, when the Delete operation is sent via the Bootstrap Interface, the Client MUST execute the following procedure.

1. If there is an Object Instance that can be accessed only by a Server of the Server Object Instance (i.e., the Server is Access Control Owner and the LWM2M Server can access the Object Instance only in an Access Control Object Instance), the Object Instance and the corresponding Access Control Object Instance MUST be deleted
2. If an Object Instance can be accessed by multiple Servers including the Server which Security Object Instance is to be deleted, then:
   - The ACL Resource Instance for the Server in the Access Control Object Instance for the Object Instance MUST be deleted
   - If the Server is the Access Control Owner of the Access Control Object Instance, then the Access Control Owner MUST be changed to another Server according to the rules below:

     The Client MUST choose the Server who has highest sum of each number assigned to an access right

(Write: 1, Delete: 1) for the Access Control Owner. If two or more Servers have the same sum, the Client MUST choose one of them as the new Access Control Owner.

3. Observation operations from the Server MUST be deleted
4. Server Object Instance MUST be deleted
5. Client MAY send "De-register" operation to the Server

Note: To monitor the change of the Access Control Owner, the Server MAY observe Access Control Owner Resource.

E.2 LWM2M Object: LWM2M Server

Description

This LWM2M Objects provides the data related to a LWM2M Server. A Bootstrap Server has no such an Object Instance associated to it.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
LWM2M Server	1	Multiple	Mandatory	urn:oma:lwm2m:oma:1

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Short Server ID	R	Single	Mandatory	Integer	1-65535		Used as link to associate server Object Instance.
1	Lifetime	RW	Single	Mandatory	Integer		s	Specify the lifetime of the registration in seconds.
2	Default Minimum Period	RW	Single	Optional	Integer		s	The default value the LWM2M Client should use for the Minimum Period of an Observation in the absence of this parameter being included in an Observation. If this Resource doesn't exist, the default value is 0.
3	Default Maximum Period	RW	Single	Optional	Integer		s	The default value the LWM2M Client should use for the Maximum Period of an Observation in the absence of this parameter being included in an Observation.

4	Disable	E	Single	Optional				If this Resource is executed, this LWM2M Server Object is disabled for a certain period defined in the Disabled Timeout Resource. After receiving "Execute" operation, LWM2M Client MUST send response of the operation and perform de-registration process, and underlying network connection between the Client and Server MUST be disconnected to disable the LWM2M Server account. After the above process, the LWM2M Client MUST NOT send any message to the Server and ignore all the messages from the LWM2M Server for the period.
5	Disable Timeout	RW	Single	Optional	Integer		s	A period to disable the Server. After this period, the LWM2M Client MUST perform registration process to the Server. If this Resource is not set, a default timeout value is 86400 (1 day).
6	Notification Storing When Disabled or Offline	RW	Single	Mandatory	Boolean			If true, the LWM2M Client stores "Notify" operations to the LWM2M Server while the LWM2M Server account is disabled or the LWM2M Client is offline. After the LWM2M Server account is enabled or the LWM2M Client is online, the LWM2M Client reports the stored "Notify" operations to the Server. If false, the LWM2M Client discards all the "Notify" operationsor temporally disables the Observe function while the LWM2M Server is disabled or the LWM2M Client is offline. The default value is true. The maximum number of storing Notification per the Server is up to the implementation.
7	Binding	RW	Single	Mandatory	String	The possible values of Resource are listed in 5.3.1.1		This Resource defines the transport binding configured for the LWM2M Client. If the LWM2M Client supports the binding specified in this Resource, the LWM2M Client MUST use that for Current Binding and Mode.
8	Registration Update Trigger	E	Single	Mandatory				If this Resource is executed the LWM2M Client MUST perform an "Update" operation with this LWM2M Server using the Current Transport Binding and Mode.

E.3 LWM2M Object: Access Control

Description

Access Control Object is used to check whether the LWM2M Server has access right for performing a operation.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
LWM2M Access Control	2	Multiple	Optional	urn:oma:lwm2m:oma:2

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Object ID	R	Single	Mandatory	Integer	1-65534		The Object ID and The Object Instance ID are applied for.
1	Object Instance ID	R	Single	Mandatory	Integer	0-65535		See Table 18: LWM2M Identifiers.
2	ACL	RW	Multiple	Optional	Integer	16-bit		Resource Instance ID MUST be the Short Server ID of a certain LWM2M Server which has an access right. Resource Instance ID 0 is for default Short Server ID. The Value of the Resource Instance contains the access rights. Setting each bit means the LWM2M Server has the access right for that operation. The bit order is specified as below. 1st lsb: R(Read, Observe, Discover, Write Attributes) 2nd lsb: W(Write) 3rd lsb: E(Execute) 4th lsb: D(Delete) 5th lsb: C(Create) Other bits are reserved for future use
3	Access Control Owner	RW	Single	Mandatory	Integer	0-65535		Short Server ID of a certain LWM2M Server. Only this LWM2M Server can manage these Resources of the Object Instance. Value MAX_ID=65535 is reserved for the Access Control Object Instances created during Bootstrap procedure.

E.3.1 Object Instance Configurations

If a new LWM2M Server Account is added when LWM2M Client has only one LWM2M Server Account, Client MUST ensure that Access Control Object Instances for every Object Instance except Security Object Instance exist. The LWM2M Client MUST create the missing Access Control Object Instances as follows:

- Access Control Owner MUST be the previously existing LWM2M Server
- Previously existing LWM2M Server MUST have full access right.

E.4 LWM2M Object: Device

Description

This LWM2M Object provides a range of device related information which can be queried by the LWM2M Server, and a device reboot and factory reset function.

Object definition

Name	Object ID	Instances	Man□at□ry	Object URN
Device	3	Single	Mandatory	urn:oma:lwm2m:oma:3

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Manufacturer	R	Single	Optional	String			Human readable manufacturer name
17	Device Type	R	Single	Optional	String			Type of the device (manufacturer specified string: e.g. smart meters / dev Class…)
1	Model Number	R	Single	Optional	String			A model identifier (manufacturer specified string)
2	Serial Number	R	Single	Optional	String			Serial Number
18	Hardware Version	R	Single	Optional	String			Current hardware version of the device
3	Firmware Version	R	Single	Optional	String			Current firmware version of the device. The Firmware Management function could rely on this resource.
19	Software Version	R	Single	Optional	String			Current software version of the device. (manufacturer specified string). On elaborated LWM2M device, SW could be split in 2 parts: a firmware one and a higher level software on top. Both pieces ofSoftware are together managed by LWM2M Firmware Update Object (Object ID 5)
22	ExtDevInfo	R	Multiple	Optional	Objlnk			Reference to external “Device” object instance containing information. For example, such an external device can be a Host Device, which is a device into which the Device containing the LWM2M client is embedded. This Resource may be used to retrieve information about the Host Device.
4	Reboot	E	Single	Mandatory				Reboot the LWM2M Device to restore the Device from unexpected firmware failure.

5	Factory Reset	E	Single	Optional				Perform factory reset of the LWM2M Device to make the LWM2M Device have the same configuration as at the initial deployment. When this Resource is executed, “De-register” operation MAY be sent to the LWM2M Server(s) before factory reset of the LWM2M Device.
6	Available Power Sources	R	Multiple	Optional	Integer	0-7		0 – DC power 1 – Internal Battery (0 or 1 only) 2 – External Battery 4 – Power over Ethernet 5 – USB 6 – AC (Mains) power 7 – Solar The same Resource Instance ID MUST be used to associate a given Power Source (Resource ID 6) with its Present Voltage (Resource ID=7) and its Present Current (Resource ID=8)
7	Power Source Voltage	R	Multiple	Optional	Integer		mV	Present voltage for each Available Power Sources Resource Instance.
8	Power Source Current	R	Multiple	Optional	Integer		mA	Present current for each Available Power Source.
9	Battery Level	R	Single	Optional	Integer	0-100	%	Contains the current battery level as a percentage (with a range from 0 to 100). This value is only valid for the Device Internal Battery if present (one Available Power Sources Resource Instance value is 1).

<table>
<tr><td>20</td><td>Battery Status</td><td>R</td><td>Single</td><td>Optional</td><td>Integer</td><td>0-6</td><td></td><td>This value is only valid for the Device Internal Battery if present (one Available Power Sources Resource Instance value is 1).
<table>
<tr><th>Battery Status</th><th>Meaning</th><th>Description</th></tr>
<tr><td>0</td><td>Normal</td><td>The battery is operating normally and not on power.</td></tr>
<tr><td>1</td><td>Charging</td><td>The battery is currently charging.</td></tr>
<tr><td>2</td><td>Charge Complete</td><td>The battery is fully charged and still on power.</td></tr>
<tr><td>3</td><td>Damaged</td><td>The battery has some problem.</td></tr>
<tr><td>4</td><td>Low Battery</td><td>The battery is low on charge.</td></tr>
<tr><td>5</td><td>Not Installed</td><td>The battery is not installed.</td></tr>
<tr><td>6</td><td>Unknown</td><td>The battery information is not available.</td></tr>
</table></td></tr>
<tr><td>10</td><td>Memory Free</td><td>R</td><td>Single</td><td>Optional</td><td>Integer</td><td></td><td>KB</td><td>Estimated current available amount of storage space which can store data and software in the LWM2M Device (expressed in kilobytes).</td></tr>
<tr><td>21</td><td>Memory Total</td><td>R</td><td>Single</td><td>Optional</td><td>Integer</td><td></td><td>KB</td><td>Total amount of storage space which can store data and software in the LWM2M Device (expressed in kilobytes).</td></tr>
</table>

11	Error Code	R	Multiple	Mandatory	Integer			0=No error 1=Low battery power 2=External power supply off 3=GPS module failure 4=Low received signal strength 5=Out of memory 6=SMS failure 7=IP connectivity failure 8=Peripheral malfunction When the single Device Object Instance is initiated, there is only one error code Resource Instance whose value is equal to 0 that means no error. When the first error happens, the LWM2M Client changes error code Resource Instance to any non-zero value to indicate the error type. When any other error happens, a new error code Resource Instance is created. This error code Resource MAY be observed by the LWM2M Server. How to deal with LWM2M Client's error report depends on the policy of the LWM2M Server.
12	Reset Error Code	E	Single	Optional				Delete all error code Resource Instances and create only one zero-value error code that implies no error.
13	Current Time	RW	Single	Optional	Time			Current UNIX time of the LWM2M Client. The LWM2M Client should be responsible to increase this time value as every second elapses. The LWM2M Server is able to write this Resource to make the LWM2M Client synchronized with the LWM2M Server.
14	UTC Offset	RW	Single	Optional	String			Indicates the UTC offset currently in effect for this LWM2M Device. UTC+X [ISO 8601].
15	Timezone	RW	Single	Optional	String			Indicates in which time zone the LWM2M Device is located, in IANA Timezone (TZ) database format.
16	Supported Binding and Modes	R	Single	Mandatory	String			Indicates which bindings and modes are supported in the LWM2M Client. The possible values of Resource are combination of "U" or "UQ" and "S" or "SQ".

E.5 LWM2M Object: Connectivity Monitoring

Description

This LWM2M Object enables monitoring of parameters related to network connectivity.
In this general connectivity Object, the Resources are limited to the most general cases common to most network bearers. It is recommended to read the description, which refers to relevant standard development organizations (e.g. 3GPP, IEEE).
The goal of the Connectivity Monitoring Object is to carry information reflecting the more up to date values of the current connection for monitoring purposes. Resources such as Link Quality, Radio Signal Strenght, Cell ID are retrieved during connected mode at least for cellular networks.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
Connectivity Monitoring	4	Single	Optional	urn:oma:lwm2m:oma:4

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Network Bearer	R	Single	Mandatory	Integer			Indicates the network bearer used for the current LWM2M communication session from the below network bearer list. 0~20 are Cellular Bearers 0: GSM cellular network 1: TD-SCDMA cellular network 2: WCDMA cellular network 3: CDMA2000 cellular network 4: WiMAX cellular network 5: LTE-TDD cellular network 6: LTE-FDD cellular network 7~20: Reserved for other type cellular network 21~40 are Wireless Bearers 21: WLAN network 22: Bluetooth network 23: IEEE 802.15.4 network 24~40: Reserved for other type local wireless network 41~50 are Wireline Bearers 41: Ethernet 42: DSL 43: PLC 44~50: reserved for others type wireline networks.
1	Available Network Bearer	R	Multiple	Mandatory	Integer			Indicates list of current available network bearer. Each Resource Instance has a value from the network bearer list.

2	Radio Signal Strength	R	Single	Mandatory	Integer		dBm	This node contains the average value of the received signal strength indication used in the current network bearer in case Network Bearer Resource indicates a Cellular Network (RXLEV range 0…64) 0 is < -110dBm, 64 is >-48 dBm). Refer to [3GPP 44.018] for more details on Network Measurement Report encoding and [3GPP 45.008] or for Wireless Networks refer to the appropriate wireless standard.
3	Link Quality	R	Single	Optional	Integer			This contains received link quality e.g., LQI for IEEE 802.15.4, (Range (0..255)), RxQual Downlink (for GSM range is 0…7). Refer to [3GPP 44.018] for more details on Network Measurement Report encoding.
4	IP Addresses	R	Multiple	Mandatory	String			The IP addresses assigned to the connectivity interface. (e.g. IPv4, IPv6, etc.)
5	Router IP Addresse	R	Multiple	Optional	String			The IP address of the next-hop IP router. Note: This IP Address doesn't indicate the Server IP address.
6	Link Utilization	R	Single	Optional	Integer	0-100	%	The average utilization of the link to the next-hop IP router in %.
7	APN	R	Multiple	Optional	String			Access Point Name in case Network Bearer Resource is a Cellular Network.
8	Cell ID	R	Single	Optional	Integer			Serving Cell ID in case Network Bearer Resource is a Cellular Network. As specified in TS [3GPP 23.003] and in [3GPP. 24.008]. Range (0…65535) in GSM/EDGE UTRAN Cell ID has a length of 28 bits. Cell Identity in WCDMA/TD-SCDMA. Range: (0..268435455). LTE Cell ID has a length of 28 bits. Parameter definitions in [3GPP 25.331].
9	SMNC	R	Single	Optional	Integer		%	Serving Mobile Network Code. In case Network Bearer Resource has 0(cellular network). Range (0…999). As specified in TS [3GPP 23.003].
10	SMCC	R	Single	Optional	Integer			Serving Mobile Country Code. In case Network Bearer Resource has 0 (cellular network). Range (0…999). As specified in TS [3GPP 23.003].

E.6 LWM2M Object: Firmware Update

Description

This LWM2M Object enables management of firmware which is to be updated. This Object includes installing firmware package, updating firmware, and performing actions after updating firmware. A reboot of the device must occur for taking into account the new successfully installed firmware.

After reboot of the device:

- the “State” Resource must be at Downloaded state (2) if the “Package” Resource contains a valid Package which has not been successfully installed yet, or at Idle state (0) otherwise.
- the Update Result must maintain the relevant value it has before Device reboot.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
Firmware Update	5	Single	Optional	urn:oma:lwm2m:oma:5

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Package	W	Single	Mandatory	Opaque			Firmware package
1	Package URI	W	Single	Mandatory	String	0-255 bytes		URI from where the device can download the firmware package by an alternative mechanism. As soon the device has received the Package URI it performs the download at the next practical opportunity.
2	Update	E	Single	Mandatory	none	no argument		Updates firmware by using the firmware package stored in Package, or, by using the firmware downloaded from the Package URI. This Resource is only executable when the value of the State Resource is Downloaded.

3	State	R	Single	Mandatory	Integer	0-3		Indicates current state with respect to this firmware update. This value is set by the LWM2M Client. 0: Idle (before downloading or after successful updating) 1: Downloading (The data sequence is on the way) 2: Downloaded 3: Updating If writing the firmware package to Package Resource is done, or, if the device has downloaded the firmware package from the Package URI the state changes to Downloaded. If writing an empty string to Package Resource is done or writing an empty string to Package URI is done, the state changes to Idle. When in Downloaded state, and the executable Resource Update is triggered, the state changes to Updating. If the Update Resource failed, the state returns at Downloaded. If performing the Update Resource was successful, the state changes from Updating to Idle.
4	Update Supported Objects	RW	Single	Optional	Boolean			If this value is true, the LWM2M Client MUST inform the registered LWM2M Servers of Objects and Object Instances parameter by sending an Update or Registration message after the firmware update operation at the next practical opportunity if supported Objects in the LWM2M Client have changed, in order for the LWM2M Servers to promptly manage newly installed Objects. If false, Objects and Object Instances parameter MUST be reported at the next periodic Update message. The default value is false.

5	Update Result	R	Single	Mandatory	Integer	0-6		Contains the result of downloading or updating the firmware 0: Initial value. Once the updating process is initiated (Download /Update), this Resource MUST be reset to Initial value. 1: Firmware updated successfully, 2: Not enough storage for the new firmware package. 3. Out of memory during downloading process. 4: Connection lost during downloading process. 5: CRC check failure for new downloaded package. 6: Unsupported package type. 7: Invalid URI 8: Firmware update failed This Resource MAY be reported by sending Observe operation.
6	PkgName	R	Single	Optional	String	0-255 bytes		Name of the Firmware Package
7	PkgVersion	R	Single	Optional	String	0-255 bytes		Version of the Firmware package

E.6.1 Firmware Update Consideration

If some Objects are not supported after firmware update, the LWM2M Client MUST delete all the Object Instances of the Objects that are not supported.

E.7 LWM2M Object: Location

Description

This LWM2M Objects provide a range of device related information which can be queried by the LWM2M Server, and a device reboot and factory reset function.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
Location	6	Single	Optional	urn:oma:lwm2m:oma:6

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Latitude	R	Single	Mandatory	String		Deg	The decimal notation of latitude, e.g. -43.5723 [World Geodetic System 1984].
1	Longitude	R	Single	Mandatory	String		Deg	The decimal notation of longitude, e.g. 153.21760 [World Geodetic System 1984].
2	Altitude	R	Single	Optional	String		m	The decimal notation of altitude in meters above sea level.

3	Uncertainty	R	Single	Optional	String		m	The accuracy of the position in meters.
4	Velocity	R	Single	Optional	Opaque		Refers to 3GPP GAD specs	The velocity of the device as defined in 3GPP 23.032 GAD specification. This set of values may not be available if the device is static.
5	Timestamp	R	Single	Mandatory	Time	0-6		The timestamp of when the location measurement was performed.

E.8 LWM2M Object: Connectivity Statistics

Description

This LWM2M Objects enables client to collect statistical information and enables the LWM2M Server to retrieve these information, set the collection duration and reset the statistical parameters.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
Connectivity Statistics	7	Single	Optional	urn:oma:lwm2m:oma:7

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	SMS Tx Counter	R	Single	Optional	Integer			Indicate the total number of SMS successfully transmitted during the collection period.
1	SMS Rx Counter	R	Single	Optional	Integer			Indicate the total number of SMS successfully received during the collection period.
2	Tx Data	R	Single	Optional	Integer		Kilo-Bytes	Indicate the total amount of data transmitted during the collection period.
3	Rx Data	R	Single	Optional	Integer		Kilo-Bytes	Indicate the total amount of data received during the collection period.
4	Max Message Size	R	Single	Optional	Integer		Byte	The maximum message size that is used during the collection period.
5	Average Message Size	R	Single	Optional	Integer		Byte	The average message size that is used during the collection period.
6	Start	E	Single	Mandatory				Reset resource 0-5 to 0 and start to collect information, If resource 8 (Collection Period) value is 0, the client will keep collecting information until resource 7 (Stop) is executed, otherwise the client will stop collecting information after specified period ended.

7	Stop	E	Single	Mandatory				Stop collecting information, but not to reset resource 0-5.
8	Collection Period	R/W	Single	Optional	Integer		Seconds	The default collection period in seconds. The value 0 indicates the collection period is not set
7	Collection Duration	RW	Single	Optional	Integer		Minutes	This is a positive integer (0 to 65535), duration can be from 1 minute to 65535 minutes (approximately 45days+) Note if the value is 0, then the collection never stops. All counters are rollover; it starts from zero when it reaches maximum value.

Appendix F. Example LWM2M Client (Informative)

This appendix defines an example LWM2M Client for a simple imaginary device with a Cellular interface including instantiated Objects and their values, which is used throughout this specification in examples. The example client has the Endpoint Name "example-client". The example device has two Server Objects (it is configured to register with two different LWM2M Servers), three accompanying Access Control Object Instances for those servers, a Device Object and a Connectivity Monitoring Object for a Cellular interface. The first Server controls the access control rights for both servers.

Object	Object ID	Object Instance ID
LWM2M Security Object[0]	0	0
LWM2M Security Object[1]	0	1
LWM2M Security Object[2]	0	2
LWM2M Server Object [1]	1	1
LWM2M Server Object [2]	1	2
Access Control Object [0]	2	0
Access Control Object [1]	2	1
Access Control Object [2]	2	2
Access Control Object [3]	2	3
Access Control Object [4]	2	4
Device Object	3	-
Connectivity Monitoring Object	4	-

Table 27: Object Instances of the example

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LWM2M Server URI	0		coap://bootstrap.example.com	Example LWM2M Bootstrap Server
Bootstrap Server	1		true	
Security Mode	2		0	PSK mode
Public Key or Identity	3		[identity string]	PSK Identity
Secret Key	4		[secret key data]	AES key
Short Server ID	10		0	unused
Client Hold Off Time	11		3600	

Table 28: LWM2M Security Object [0]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LWM2M Server URI	0		coap://server1.example.com	Example LWM2M Server 1
Bootstrap Server	1		false	
Security Mode	2		0	PSK mode
Public Key or Identity	3		[identity string]	PSK Identity
Secret Key	4		[secret key data]	AES key
Short Server ID	10		101	
Client Hold Off Time	11		0	unused

Table 29: LWM2M Security Object [1]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LWM2M Server URI	0		coap://server2.example.com	Example LWM2M Server 2
Bootstrap Server	1		false	
Security Mode	2		0	PSK mode
Public Key or Identity	3		[identity string]	PSK Identity
Secret Key	4		[secret key data]	AES key
Short Server ID	5		102	
Client Hold Off Time	6		0	unused

Table 30: LWM2M Security Object [2]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Short Server ID	0		101	Example LWM2M Server 1
Lifetime	1		86400	
Default Minimum Period	2		300	
Default Maximum Period	3		6000	
DisableTimeout	5		86400	
Notification Storing When Disabled or Offline	6		True	
Binding Preference	7		U	UDP binding preference

Table 31: LWM2M Server Object [1]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Short Server ID	0		102	Example LWM2M Server 2
Lifetime	1		86400	
Default Minimum Period	2		60	
Default Maximum Period	3		6000	
DisableTimeout	5		86400	
Notification Storing When Disabled or Offline	6		False	
Binding Preference	7		UQ	UDP with Queuing binding preference

Table 32: LWM2M Server Object [2]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		1	LWM2M Server Object
Object Instance ID	1		0	
ACL	2	101	0b0000000000001111	Server 1 has all access rights (R, W, E, D). Note that the Resource Instance ID indicates the Short Server ID.
Access Control Owner	3		101	Server 1 controls this Object Instance's access rights.

Table 33: Access Control Object [0] (for the LWM2M Server Object)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		1	LWM2M Server Object
Object Instance ID	1		1	
ACL	2	102	0b0000000000001111	Server 2 has all access rights (R, W, E, D). Note that the Resource Instance ID indicates the Short Server ID.
Access Control Owner	3		102	Server 2 controls this Object Instance's access rights.

Table 34: Access Control Object [1] (for the LWM2M Server Object)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		3	Device Object
Object Instance ID	1		0	
ACL	2	101	0b0000000000001111	Server 1 has all access rights (R, W, E, D). Note that the Resource Instance ID indicates the Short Server ID.
ACL	2	102	0b0000000000000001	Server 2 has read-only access rights. Note that the Resource Instance ID indicates the Short Server ID.
Access Control Owner	3		101	Server 1 controls this Object Instance's access rights.

Table 35: Access Control Object [2] (for the Device Object)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		4	Connectivity Monitoring Object
Object Instance ID	1		0	
ACL	2	101	0b0000000000000001	Server 1 has read-only access rights. Note that the Resource Instance ID indicates the Short Server ID.
ACL	2	0	0b0000000000000001	The other Servers except Server 1 have read-only access rights. Note that this Resource Instance ID indicates the default Short Server ID.
Access Control Owner	3		101	Server 1 controls this Object Instance's access rights.

Table 36: Access Control Object [3] (for the Connectivity Monitoring Object)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		5	Firmware Update Object
Object Instance ID	1		65535	Irrelavent
ACL	2	101	0b0000000000010000	Server 1 can create Firmware Update Object Instance
Access Control Owner	3		65535	This Object Instance must be managed by Bootstrap Interface

Table 37: Access Control Object [4] (for the Firmware Update Object)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Manufacturer	0		Open Mobile Alliance	
Model Number	1		Lightweight M2M Client	
Serial Number	2		345000123	
Firmware version	3		1.0	
Available Power Sources	6	0	1	Internal Battery
Available Power Sources	6	1	5	USB
Power Source Voltage	7	0	3800	3.8V battery
Power Source Voltage	7	1	5000	USB VBUS
Power Source Current	8	0	125	125mA
Power Source Current	8	1	900	USB 900mA
Battery level	9		100	
Memory free	10		15	15 kB of free memory
Error code	11	0	0	No errors
Current Time	13		1367491215	May 2nd, 2013 at 11:42 AM

				GMT
UTC Offset	14		+02:00	UTC+2 (CET)
Supported Binding and Modes	15		U	UDP binding

Table 38: Device Object

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Network Bearer	0		0	GSM Bearer
Available Network Bearer	1		0	GSM Bearer
Radio signal strength	2		92	RSSI in dBm
Link Quality	3		2	RxQual Downlink
IP Addresses	4	0	192.168.0.100	
Parent IP Addresses	5	0	192.168.1.1	
Link Utilization	6		5	%
APN	7	0	internet	

Table 39: Connectivity Monitoring Object

Appendix G. Storage of LWM2M Bootstrap Information on the Smartcard (Normative)

This appendix aims at specifying the storage mechanism of Bootstrap Information on UICC Smartcard platform type [ETSI TS 102.221] activated in 3G mode.

Note: There is no rational to equip LWM2M device with 2G-only Smart Card.

G.1 File structure

The information format is based on [PKCS#15] specification. The Bootstrap data is located under the PKCS#15 directory allowing the card issuer to decide the identifiers and the file locations. The smartcard operations that are relevant include:

- Application selection
- Cardholder verification
- File access (select file, read, write)

The [PKCS#15] specification defines a set of files. Within the PKCS#15 application, the starting point to access these files is the Object Directory File (ODF). The EF (ODF) contains pointers to other directory files. These directory files contain information on different types of objects (authentication objects, data objects, etc). For the purpose of Bootstrap data, EF (ODF) MUST contain the EF Record describing the DODF-bootstrap. The EF (ODF) is described in Appendix G.3.1 and [PKCS#15].

EF (ODF) contains pointers to one or more Data Object Directory Files (DODF) in priority order (i.e. the first DODF has the highest priority). Each DODF is regarded as the directory of data objects known to the PKCS#15 application. For the purposes of LWM2M bootstrapping, EF (DODF-bootstrap) contains pointer to the Bootstrap data, namely LWM2M_Bootstrap File. The EF (DODF-bootstrap) is described in Appendix G.3.2 and [PKCS#15].

The provisioning files are stored as PKCS#15 opaque data objects.

The support of smartcard Bootstrap data will be indicated by the presence in the EF DIR (see [ETSI TS 102.221]) of an application template as defined here after.

The RECOMMENDED format of EF (DIR) is a linear fixed record in order to be in line with [ETSI TS 102.221].

EF (DIR) MUST contain the application template used for a PKCS#15 application as defined in [PKCS#15]. Application template MUST consist of Application identifier (tag 0x4F) and Path (tag 0x51) information.

The EF (ODF) and EF (DODF-bootstrap) MUST be used by the Device to determine the path of the LWM2M_Bootstrap file.

UICC Smartcard platforms can support two modes of activation: 2G and 3G. In the context of LWM2M, for Device simplification, UICC MUST be activated in 3G Mode

UICC smartcard platform activated in a 3G mode has the physical and logical characteristics according to [ETSI TS 102.221]. In that case, smartcard operations for accessing the Bootstrap data are specified in Appendix G.2.

G.2 Bootstrap Information on UICC (Activated in 3G Mode)

G.2.1 Access to the file structure

To select the PKCS#15 application, the Device:

- MUST evaluate the PKCS#15 application template – i.e. PKCS#15 AID - present in the EF (DIR),
- MUST open a logical channel using UICC Command MANAGE CHANNEL as specified in [ETSI TS 102.221],
- MUST select the PKCS#15 ADF using the PKCS#15 AID as parameter of the UICC Command SELECT, using direct application selection as defined in [ETSI TS 102.221].

LWM2M_Bootstrap file will be located under the PKCS#15 ADF.

G.2.2 Files Overview

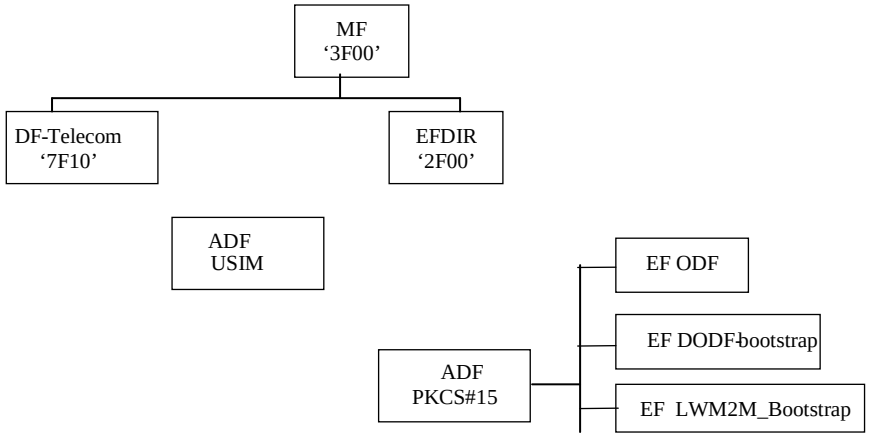


Figure 27: 3G UICC File Structure and Bootstrap data location

G.2.3 Access Method

UICC Commands Read Binary and Update Binary, as defined in [ETSI TS 102.221], are used to access bootstrap data.

G.2.4 Access Conditions

The Device is informed of the access conditions of provisioning files by evaluating the "private" and "modifiable" flags in the corresponding DODF-bootstrap files structure.

In the case where one of the above mentioned flag is set, cardholder verification is required. The Device must evaluate the PIN references that must be verified as defined in [ETSI TS 102.221] (ie evaluate FCP)

G.2.5 Requirements on the 3G UICC

To retrieve the Bootstrap Information from the 3G UICC, the Device MUST perform the following steps:

- Select PKCS#15 file structure as specified in G.2.1.
- Read ODF to locate the DODF-bootstrap,
- Read DODF-bootstrap to locate the LWM2M_Bootstrap file,
- Read the LWM2M_Bootstrap file

G.3 Files Description

All files defined are binary files as defined in [**ETSI TS 102.221**]. These files are read and updated using 3G UICC Commands related to the application they belong to.

G.3.1 Object Directory File, EF ODF

The mandatory Object Directory File (ODF) ([PKCS#15], Section 5.5.1) contains pointers to other EFs, each one containing a directory of PKCS#15 objects of a particular class (e.g. DODF-bootstrap). The File ID is specified in [PKCS#15]. The card issuer decides the file size. The EF (ODF) can be read but it MUST NOT be modifiable by the user.

The EF (ODF) is described below:

Identifier: default 0x5031, see [PKCS#15]	Structure: Binary	Mandatory
File size: decided by the card issuer	Update activity: low	
Access Conditions: READ ALW UPDATE ADM INVALIDATE ADM REHABILITATE ADM		
Description		
See [PKCS#15]		

G.3.2 Bootstrap Data Object Directory File, EF DODF-bootstrap

This Data Object Directory File provisioning contains directories of provisioning data objects ([PKCS#15], Section 6.7) known to the [PKCS#15] application.

The File ID is described in the EF (ODF). The file size depends on the number of provisioning objects stored in the smartcard. Thus, the card issuer decides the file size.

Identifier: 0x6430, See ODF	Structure: Binary	Mandatory
File size: decided by the card issuer	Update activity: low	
Access Conditions: READ ALW or Universal / application / Local PIN (UICC, See Appendix D.2) UPDATE ADM INVALIDATE ADM REHABILITATE ADM		
Description		
See hereafter and [PKCS#15]		

The EF (DODF-bootstrap) MUST contain information on provisioning objects:

- Readable label describing the provisioning document (CommonObjectAttributes.label). The ME could display this label to the user.
- Flags indicating whether the provisioning document is private (i.e., is protected with a PIN) and/or modifiable (CommonObjectAttributes.flags). The card issuer decides whether or not a file is private (it does not need to be if it does not contain any sensitive information)
- Object identifier indicating a LWM2M boostrap Object and the type of the provisioning object (CommonDataObjectAttributes.applicationOID)
- Pointer to the contents of the provisioning document (Path.path)

G.3.3 EF LWM2M_Bootstrap

Only the card issuer can modify EF LWM2M_Bootstrap

Identifier: See DODF	Structure: Binary	Optional
File size: decided by the card issuer	Update activity: low	
Access Conditions: READ ALW or Universal / application / Local PIN (UICC, See Appendix D.2) UPDATE ADM INVALIDATE ADM REHABILITATE ADM		
Description		
Contains Bootstrap data (encapsulated LWM2M Objects)		

This file size is limited to 32KB; the effective file size, in Bytes, is accessible from the File header.

In this file, the Bootstrap data relies on LWM2M TLV Data format specification.

The LWM2M specification already describes the TLV format for coding multiples instances and Resources of a given Object (§6.3.3)., this section will only detailled how storing a collection of LWM2M Objects in this file; each Object being coded as a simple TLV with LWM2M Object ID as the tag, a LWM2M-TLV coding the Object Instances as the TLV payload, and the TLV length being the size of the payload (LWM2M-TLV of the Object Instances).

Additionally, this Bootstrap data will have a 2 Byte header indicating the number of Objects contained in that file and another 2 Bytes for indicating the size of the payload (size of the collection of LWM2M Objects).

Using a BNF-like description:

```
<bootstrap_data>              ::= <number of objects> <size> <collection_of_lwm2m_objects>
<number of Objects>           ::= HWORD
<size>                       ::= HWORD
<collection_of_lwm2m_objects> ::= <single_lwm2m_object>*
<single_lwm2m_object>  ::=  <lwm2m_object_ID> <length_of_object> <lwm2m_object_instances>
<lwm2m_object_ID>             ::=  HWORD
<length_of_object>            ::=  HWORD
<lwm2m_object_instances>      ::= TLV data format as described in §6.3.3
HWORD                         ::= %x00-FFFF
```

In reading and processing the data of this file, the LWM2M Client is then able to be configured with the Bootstrap Information and thus to access the LWM2M Server(s)

Appendix H. Secure channel between Smartcard and LWM2M Device Storage for secure Bootstrap Data provisioning (Normative)

During LWM2M Bootstrap procedure, sensitive data have to be provisioned in LWM2M Device.

When Bootstrap information comes from Smartcard, a secure channel SHOULD be established. When required this secure channel MUST follow the following procedure based on [GLOBALPLATFORM] [GP SCP03] which is illustrated below. The Bootstrap information will be retrieved from Smartcard as described in Appendix F of this document but in including the channel securisation.

Pre-requisite: the Smartcard and the LWM2M device have to share the same static Keys KEY_ENC, KEY_MAC, KEY_DEK as specified in [GLOBALPLATFORM] [GP SCP03]

These keys are provisioned in the devices in using out-of-band methods.

The steps for the secure transfer are the following and are illustrated below (Figure 28):

- The PKSC#15 application used for transferring the Bootstrap information is selected
- Secure channel (mutual authentication) is established
- PKCS#15 flow as described in Appendix F takes place for selecting and transferring the Bootstrap file from Smartcard to the device: the sensitive Bootstrap data are transferred crypted.

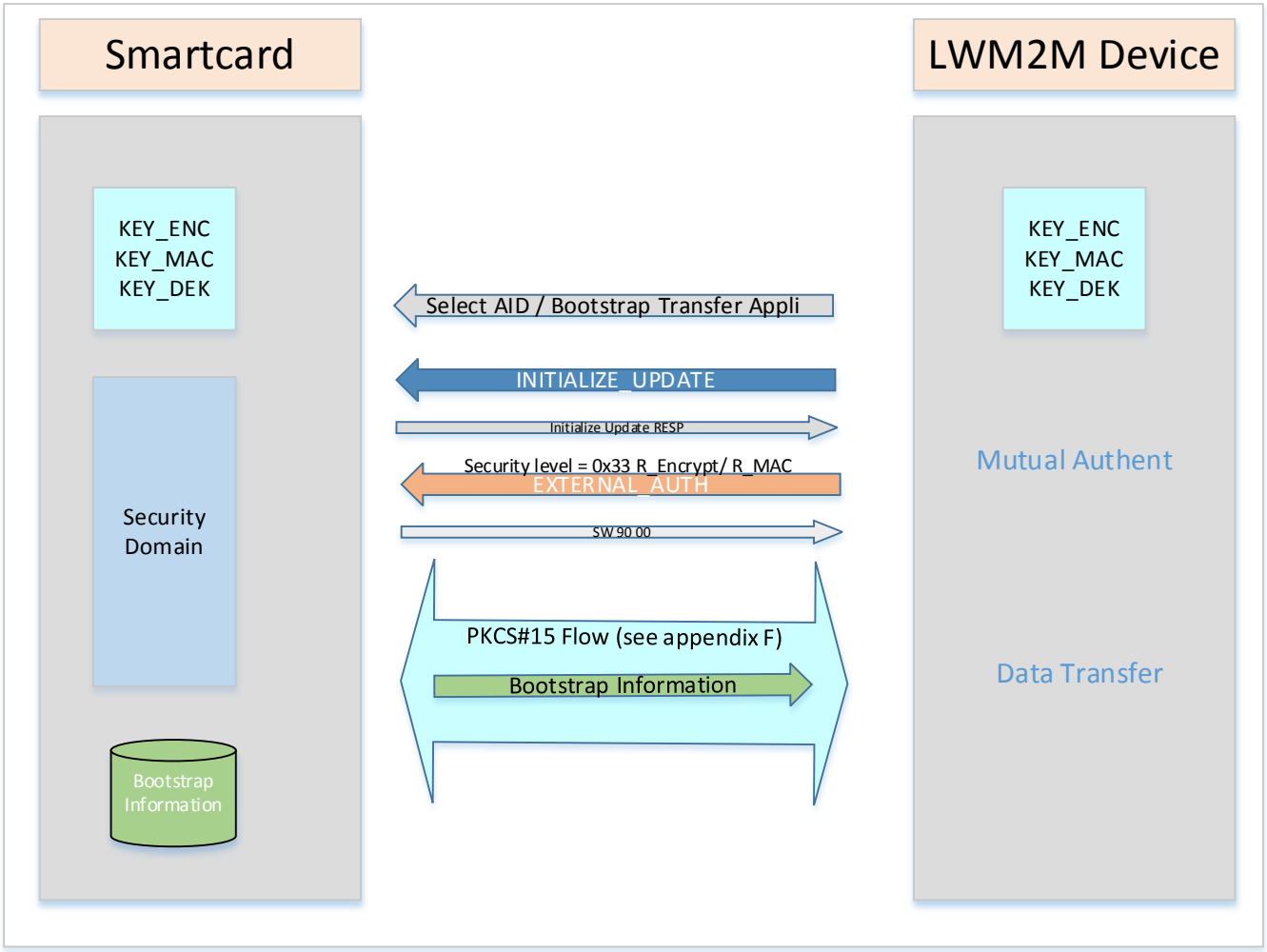


Figure 28: Bootstrap Infromation transfer from Smartcard to LWM2M Device using Secure channel according to [GLOBALPLATFORM] [GP SCP03] [GP AMD_A]

Note 1: The INITIALIZE_UPDATE specifies the logical channel to use (CLA: 80H / 83H)

Note 2: The security level (P1) of the EXTERNAL_AUTH command is C-DECRYPTION, R-ENCRYPTION, C-MAC and R-MAC (P1=0x33)

Appendix I. MIME media types

I.1 Media-Type Registration Request for application/vnd.oma.lwm2m+tlv

This section provides the registration request, as per [RFC6838], to be submitted to IANA.

Type name: application

Subtype name: vnd.oma.lwm2m+tlv

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations:

OMA LWM2M data is passive, meaning it does not contain executable or active content which may represent a security threat. The OMA LWM2M TLV format does not contain fields which are confidential. The usage of the LWM2M TLV format may be vulnerable to attacks modifying or spoofing the content of this format. TheOMA LWM2M protocol uses source authentication and integrity protection.

Interoperability considerations:

This content type carries OMA LWM2M data model serialization within the scope of the OMA LWM2M enabler. The OMA LWM2M enabler specification includes static conformance requirements for this content.

Published specification:

OMA LWM2M 1.0 Technical Specification – especially section 6.3.3. Available from http://www.openmobilealliance.org

Applications, which use this media type:

OMA LWM2M

Fragment identifier considerations: none

Additional information:

- Deprecated alias names for this type: none
- Magic number(s): none
- File extension(s): none
- Macintosh File Type Code(s): none

Intended usage: Limited use.

Only for usage with OMA LWM2M, which meet the semantics given in the mentioned specification.

Restriction on usage: no

Person & email address to contact for further information:

John Mudge, helpdesk@omaorg.org

Author/Change controller:

Open Mobile Naming Authority (OMNA).

OMA-OMNA@mail.openmobilealliance.org

Provisional registration (standards tree only): LWM2M+TLV

I.2 Media-Type Registration Request for application/vnd.oma.lwm2m+json

This section provides the registration request, as per [RFC6838], to be submitted to IANA.

Type name: application

Subtype name: vnd.oma.lwm2m+json

Required parameters: none

Optional parameters: none

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC7159].

Security considerations:

OMA LWM2M data is passive, meaning it does not contain executable or active content which may represent a security threat. The OMA LWM2M JSON format does not contain fields which are confidential. The usage of the LWM2M JSON format may be vulnerable to attacks modifying or spoofing the content of this format. The OMA LWM2M protocol uses source authentication and integrity protection.

Interoperability considerations:

This content type carries OMA LWM2M data model serialization within the scope of the OMA LWM2M enabler. The OMA LWM2M enabler specification includes static conformance requirements for this content.

Published specification:

OMA LWM2M 1.0 Technical Specification – especially section 6.3.4. Available from http://www.openmobilealliance.org

Applications, which use this media type:

OMA LWM2M

Fragment identifier considerations: none

Additional information:

- Depecated alias names for this type: none
- Magic number(s): none
- File extension(s): none
- Macintosh File Type Code(s): none

Intended usage: Limited use.

Only for usage with OMA LWM2M, which meet the semantics given in the mentioned specification.

Restriction on usage: no

Person & email address to contact for further information:

John Mudge, helpdesk@omaorg.org

Author/Change controller:

Open Mobile Naming Authority (OMNA).

OMA-OMNA@mail.openmobilealliance.org

Provisional registration (standards tree only): LWM2M+JSON