Lightweight Machine to Machine Technical Specification

Approved Version 1.0 – 08 Feb 2017

Open Mobile Alliance

OMA-TS-LightweightM2M-V1_0-20170208-A

Contents

Figures

Tables

1. Scope

This document specifies version 1.0 of the Lightweight Machine-to-Machine (LwM2M) protocol. This Lightweight M2M 1.0 enabler introduces the following features:

- Simple resource model with the core set of objects and resources defined in this specification. The full list of registered objects can be found at [OMNA].
- Operations for creation, update, deletion, and retrieval of resources.
- Asynchronous notifications of resource changes.
- Support for several serialization formats, namely TLV, JSON, Plain Text and binary data formats and the core set of LightweightM2M Objects.
- UDP and SMS transport support.
- Communication security based on the DTLS protocol supporting different types of credentials.
- Queue Mode offers functionality for a LwM2M Client to inform the LwM2M Server that it may be disconnected for an extended period and when it becomes reachable again.
- Support for use of multiple LwM2M Servers.
- Provisioning of security credentials and access control lists by a dedicated LwM2M bootstrap-server.

2. References

2.1 Normative References

[3GPP-TS_23.003]	3GPP TS 23.003 "Numbering, addressing and identification"
[3GPP-TS_23.032]	3GPP TS 23.032 "Universal Geographical Area Description (GAD)"
[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_24.008]	3GPP TS 24.008 "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3"
[3GPP-TS_25.331]	3GPP TS 25.331 "Radio Resource Control (RRC); Protocol specification"
[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
[CoAP_Blockwise]	C. Bormann, Z. Shelby, "Block-wise transfers in CoAP", IETF RFC 7959.
[CoAP-EST]	S. Kumar, P. van der Stok, "EST based on DTLS secured CoAP (EST-coaps)", draft-vanderstok-core-coap-est-00, October, 2016
[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 1.13, Open Mobile Alliance™, OMA-IOP-Process-V1_13, 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", IETF RFC 7641.
[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

[RFC5280] D. Cooper, et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[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

[RFC5958] S. Turner, "Asymmetric Key Packages", RFC 5958, August 2010.

[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.

[RFC7292] K. Moriarty, et al., "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, July 2014.

[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

[DYNAMIC LINK] "Dynamic Resource Linking for Constrained RESTful Environments", Z.Shelby, Z.Vial, M.Koster, C.Groves, Oct 2016, draft-ietf-core-dynlink-01

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

[ISO/IEC18031:2011] ISO, "ISO/IEC 18031:2011: Information technology -- Security techniques -- Random bit generation", November 2011, available at http://www.iso.org/iso/catalogue_detail.htm?csnumber=54945

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

[OMNA] "OMNA Lightweight M2M (LwM2M) Object & Resource Registry", URL:http://www.openmobilealliance.org/

[RESOURCE DIRECTORY] "CoRE Resource Directory", Z.Shelby, M.Koster, C.Bormann, P.Van Der Stok Oct 2016, draft-ietf-core-resource-directory-09

[RFC3986] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[RFC4086] D. Eastlake, J. Schiller, S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.

[RFC6698] P. Hoffman, J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC7459] "Representation of Uncertainty and Confidence in the Presence Information Data Format Location Object (PIDF-LO)", M. Thomson, J. Winterbootom, February 2015. URL:https://tools.ietf.org/html/rfc7459

[SMS-DTLS] "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", H. Tschofenig, T. Fossati, July 2016, URL:http://www.ietf.org/rfc/rfc7925.txt

[SP800-90A] Elaine Barker, John Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication 800-90A", Revision 1, June 2015, available at http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

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

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

3.3 Abbreviations

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 LwM2M Client and the M2M service, platform or application acts as the LwM2M Server. The LwM2M Enabler has two components, LwM2M Server and LwM2M Client. Four interfaces are designed between these two components as shown below:

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

This architecture is shown in Figure 1. The LwM2M Enabler uses the Constrained Application Protocol (CoAP) with UDP and/or SMS bindings. Datagram Transport Layer Security (DTLS) provides security for UDP transport layer. The LwM2M Enabler protocol stack is shown in Figure 2.

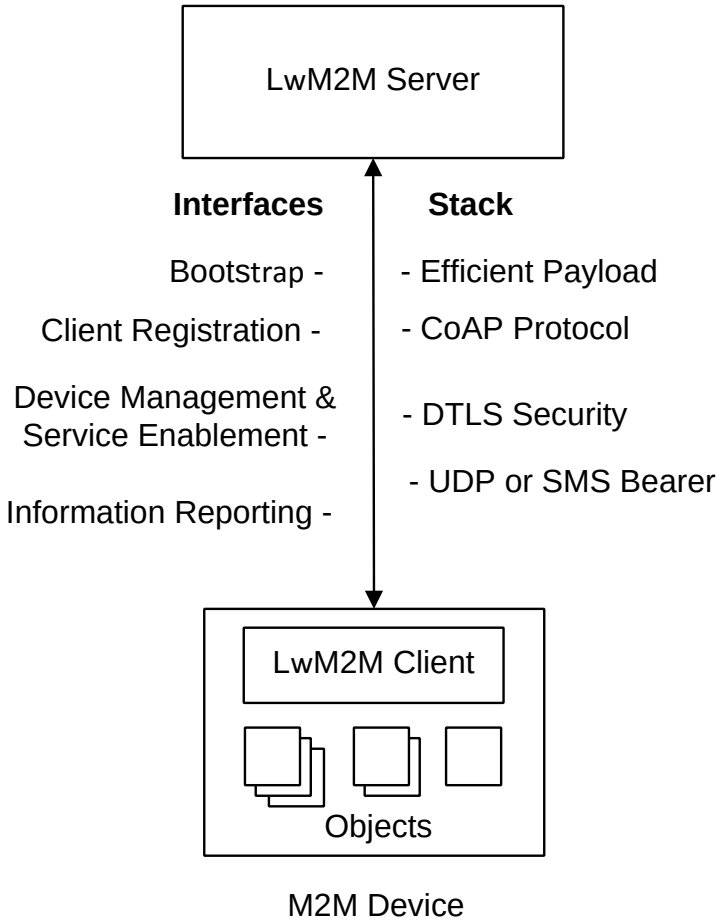


Figure 1: The overall architecture of the LwM2M Enabler

Figure 1: The overall architecture of the LwM2M Enabler.

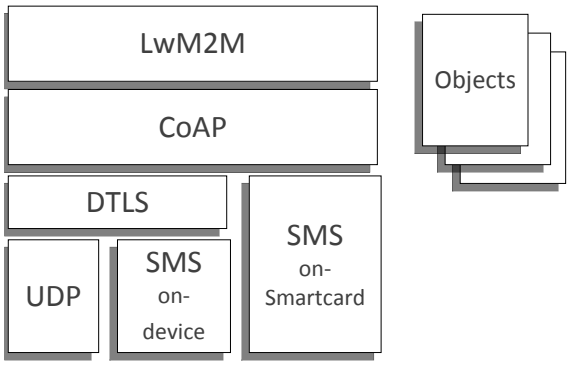


Figure 2: The protocol stack of the LwM2M Enabler.

4.1 Version 1.0

Version 1.0 of LwM2M brings in basic enablers needed, the following are the list of object enablers defined as part of core TS.

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

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 8 Transport Layer Bindings and Encodings.

Figure 3 shows the operation model for interface "Bootstrap". For this interface, the operations are an uplink operation named "Bootstrap-Request" and downlink operations named "Discover", "Write", "Delete" and "Bootstrap-Finish". 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).

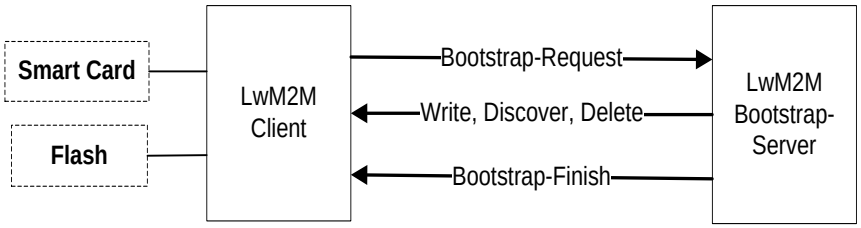


Figure 3: Bootstrap

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".

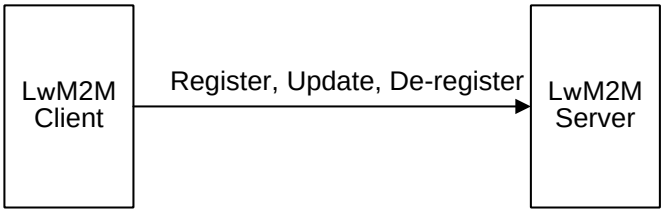


Figure 4: Client Registration

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 by the 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 used to create or delete Instances.

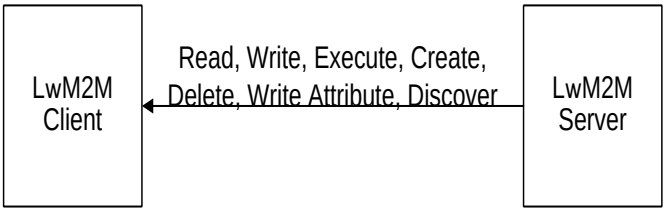


Figure 5: Device Management and Service Enablement

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.

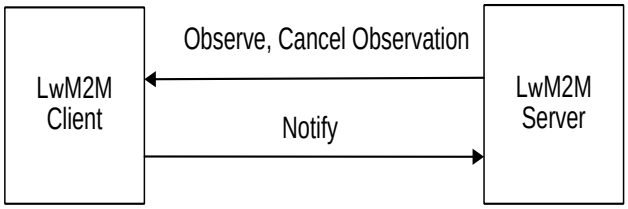


Figure 6: Information Reporting

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

Interface	Direction	Operation
Bootstrap	Uplink	Bootstrap-Request
Bootstrap	Downlink	Write, Discover, Delete, Bootstrap-Finish
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. The value of an Attribute is LwM2M Server specific. These attributes can fulfil 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 the LwM2M entity a given Attribute is attached to, the value of such an Attribute can be set at various levels: Object, Object Instance, Resource levels. Additionally, precedence rules apply when the same Attribute receives a value at different levels.

Several rules govern usage of LwM2M Attributes

- The value of an O-Attribute MAY only be set at the Object level.
- The value of an OI-Attribute MAY be set at the Object Instance level, and at the Object level.

  precedence rules:
  - Rule 1: When set at both levels, the value of the OI-Attribute set at Object Instance level will prevail.
  - Rule 2: When the Attribute value is set at 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 set at 3 different levels: the Resource level, the Object Instance level and the Object level.

  precedence rules:
  - Rule 3: When set at 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 set at the Object Instance level, the scope of an R-Attribute value extends to all the Resources of that Object Instance as long as the Rule 3 is respected.
  - Rule 5: When set at the Object level, the scope of an R-Attribute value extends to all the resources of any Instance of that Object, as long as the Rule 4 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 value of the Attribute MAY be set.
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 information
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> or “-” or “ ”
Value Type	Data Type (Refer Appendix C)
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 to 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 payload response to a “Discover” command is a list of application/link-format CoRE Links [RFC6690] which will include 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	Support Required	Access Mode	Value Type	Default Value	Applicability	Notes
Dimension	dim	Resource	Resource	YES (Client)	R	Integer [0:255]	-	Multiple Resource	Number of Instantiations for a Multiple Resource
Object Version	ver	Object	Object	YES (Server) YES (Client) when able to support Objects in version > 1.0	R	String	1.0 (Initial Version)		Provide the version of the associated Object. The rules governing the usage of this parameter are specified in Section 6.2 and Table 19.

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 "Minimum 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 Object Instance Object	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 Object Instance Object	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 "Maximum 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	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	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	st	Resource	Resource	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*"st" values <"gt" value)

Table 4: <NOTIFICATION> class Attributes

Examples illustrating Attributes setting are provided in Section 8.2.5 (Device Management & Service Enablement Interface).

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.

During the Bootstrap Phase, the Client MAY ignore requests and flush all pending responses not related to the Bootstrap sequence. There are four bootstrap modes supported by the LwM2M Enabler:

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

The last two Bootstrap modes require the help of a LwM2M Bootstrap-Server to achieve the ultimate goal to connect a LwM2M Client to their LwM2M Server(s).

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 LwM2M Bootstrap-Server

The LwM2M Bootstrap-Server is used to provision the LwM2M Client with the information required to contact the LwM2M Server(s).

In order for the LwM2M Client and the LwM2M Bootstrap-Server to establish a connection on the Bootstrap Interface, either in Client Initiated Bootstrap mode or in Server Initiated Bootstrap mode, the LwM2M Client MUST have a LwM2M Bootstrap-Server Account pre-provisioned in it.

5.2.2 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 5.2.4 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 5.2.4.

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.

Note that according to the LwM2M Server Account definition, a usual LwM2M Server Account is composed of a Security Object Instance and a Server Object Instance which are paired by sharing (respectively in Resource 10 and Resource 0 of that Objects) the same Short Server ID; a Short Server ID being unique in the LwM2M Client.

The LwM2M Server Bootstrap Information MAY additionally contain further Object Instances (e.g., Access Control, Connectivity Monitoring 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 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 Monitoring 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 completion of Bootstrap Sequence specified in 5.2.4.

Please note that the LwM2M Client MUST accept Bootstrap Information sent via Bootstrap Interface without applying access control as specified in Section 7.3.2 Authorization.

5.2.3 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

5.2.3.1 Factory Bootstrap

In this mode, the LwM2M Client has been configured with the necessary Bootstrap Information prior to deployment of the device.

5.2.3.2 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 G. When the bootstrap data retrieval is successful, the LwM2M Client MUST process the bootstrap data from the Smartcard and SHOULD apply the Bootstrap Information to its configuration to enhance security benefits.

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 H.

In this Bootstrap mode, the LwM2M Client MUST also ensure that the bootstrap data previously retrieved from the Smartcard is unchanged within the Smartcard. If bootstrap data is changed, and if the previous Bootstrap Information was applied from Smartcard, this previous Bootstrap Information MUST be disabled in the LwM2M Client and the LwM2M Client SHOULD apply the new Bootstrap Information from Smartcard to its configuration.

If the Smartcard is disabled (e.g., removing the Smartcard) then 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 (5.2.5) 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].

5.2.3.3 Client Initiated Bootstrap

As defined in Section 5.2.4 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 a LwM2M Bootstrap-Server Account preloaded in the LwM2M Client.

The minimum information that needs to be preloaded, is the security credentials required for a secure DTLS connection to the LwM2M Bootstrap-Server.

The figure below depicts the Client Initiated Bootstrap flow.

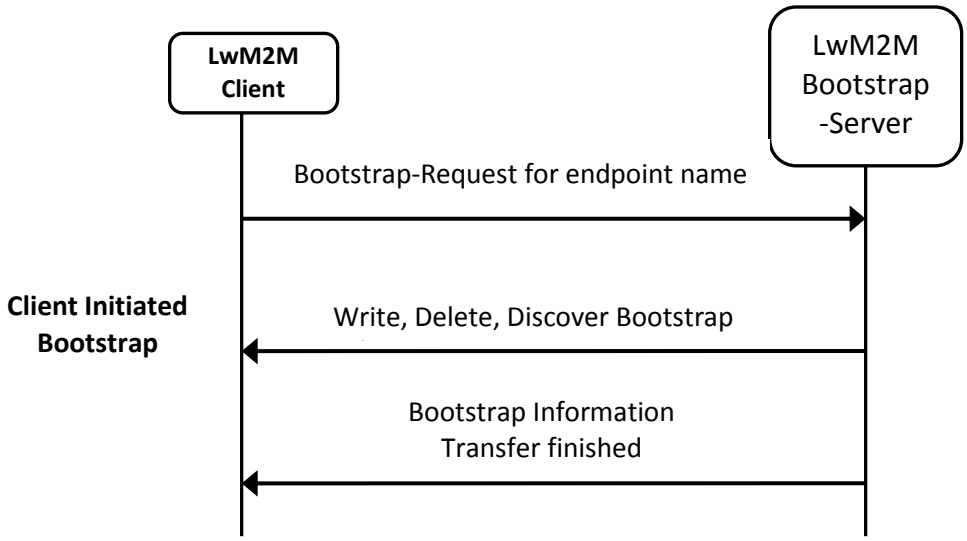


Figure 7: Procedure of Client Initiated Bootstrap

Step #0: Bootstrap-Request to bootstrap URI

The LwM2M Client sends a "Bootstrap-Request" 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 to allow the LwM2M Bootstrap-Server to provision the proper Bootstrap Information for the LwM2M Client.

Step #1: Configure Bootstrap Information

The LwM2M Bootstrap-Server configures the LwM2M Client with the Bootstrap Information using the "Write" and/or "Delete" operation.

This Bootstrap mode 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 #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.

In case the LwM2M Client didn't receive such a Finish Bootstrap Indication in a certain period (EXCHANGE_LIFETIME) after the last received Bootstrap-Server command, the LwM2M Client MUST consider the Bootstrap procedure is failed.

EXCHANGE_LIFETIME is defined in RFC 7252 [CoAP].

Step #3: Clean-up after successful Bootstrapping

Successful Bootstrapping means that the Bootstrap-Finish command has been received by the LwM2M Client, AND the loaded Configuration is considered consistent by the LwM2M Client, i.e. the Bootstrap-Finish response code sent to the Bootstrap-Server is 2.04 (Changed); if these two conditions are not met, it is an unsuccessful Bootstrapping, and in that case the Bootstrap-Server Account MUST retain the values it had before the unsuccessful Bootstrapping sequence starts; consequently the following specification of this step#3 doesn't apply.

If the Bootstrap-Server Account Timeout Resource is instantiated in the Security Object Instance of the Bootstrap-Server, the LwM2M Client MUST purge the LwM2M Bootstrap-Server Account after the expiration time provided by the value of this Resource. If this Resource is not instantiated or its value is set to 0, the Bootstrap-Server Account lifetime is infinite (Section E.1 Security Object).

High entropy keys that are unique per device SHOULD be used for the LwM2M Bootstrap-Server Account. In that case the LwM2M Bootstrap-Server Account SHOULD be kept after bootstrapping i.e. the Bootstrap-Server Account Timeout Resource may be set to 0, or may stay not instantiated.

In case the Bootstrap-Server Account has to be replaced, the replacement and the purge of the previous Bootstrap-Server Account MUST properly take place before the Client sends the Bootstrap-Finish response message back to the Bootstrap-

Server; otherwise a "4.06 Not Acceptable" Response code MUST be returned, and the previous Bootstrap-Server Account is still the only one active.

Note: If the original LwM2M Bootstrap-Server Account is purged from the device, and a new LwM2M Bootstrap-Server Account has NOT been 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.

5.2.3.4 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 "Bootstrap-Request" 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.

The Server Initiated Bootstrap mode requires a LwM2M Bootstrap-Server Account preloaded in the LwM2M Client. The minimum information that needs to be preloaded, is the security credentials required for a secure DTLS connection to the LwM2M Bootstrap-Server.

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 by strictly replicating the procedures of the step#1 step#2 and step#3 specified in the LwM2M Client Initiated Bootstrap mode.

The figure below depicts the Server Initiated Bootstrap flow.

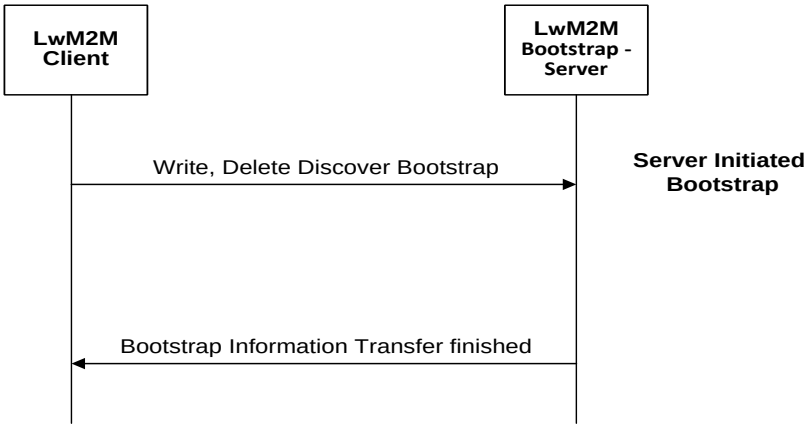


Figure 8: Procedure of Server Initiated Bootstrap

5.2.4 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.

5.2.5 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.

5.2.6 Bootstrap and Configuration Consistency

When a Bootstrap Information is loaded in the LwM2M Client, any detected inconsistency MUST be reported in sending an error response code to the BOOTSTRAP-FINISH command.

As an example, during a Bootstrap phase, a Multi-Servers Configuration is loaded in a LwM2M Client with the associated Instances of the Access Control Object, while this particular LwM2M Client is not supporting the Access Control Object itself (Single Server context support only). In that case, the client is not able to find a satisfactory consistent configuration by itself, so that the Bootstrap sequence cannot be ended successfully.

On receipt of the error response code to the BOOTSTRAP-FINISH command, the Bootstrap-Server MAY take corrective actions before issuing a new BOOTSTRAP-FINISH command.

As specified in Section 5.2.3 "Bootstrap Modes", the LwM2M Client MUST consider the Bootstrap procedure is failed when the LwM2M Client didn't receive a BOOTSTRAP-FINISH command in a certain period (EXCHANGE_LIFETIME).

5.2.7 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).

5.2.7.1 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.

5.2.7.2 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	-	-

5.2.7.3 Bootstrap DISCOVER

The “Discover” operation in Bootstrap Interface is different from the “Discover” operation in Device Management and Service Enablement interface.

The “Discover” operation on the Bootstrap Interface is used to discover which LwM2M Objects and Object Instances are supported by a certain LwM2M Client. In particular, the Security Object Instances (ID:0) are reported, while they are not accessible in Device Management and Service Enablement Interface. This operation is useful to clean-up or to update a LwM2M Client configuration (e.g., adding (removing) a LwM2M Server Account to (from) the LwM2M Client configuration).

The returned payload is a list of application/link-format CoRE Links [RFC6690] containing the LwM2M Enabler Version and for each targeted Object - in addition with the LwM2M Enabler Version and the Object Versions if they are required (see section 6.2 “Object Versioning” and Table 3 of this document <PROPERTIES> Class Attributes) - a sub-list of its Object Instances. In order to fully identify the Server supported by the Client, the Object Instances of the Server Account(s) (Object ID:0, Object ID:1) also includes the associated Short Server ID in the parameters list.

The Bootstrap-Server Security Object Instance doesn’t include a Short Server ID in its parameter list, since it has none associated with.

The targeted URI ‘/’ is accepted in place of the Object ID URI, for informing the Client to report all existing Object Instances.

The “Discover” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	No	-	Indicates the Object. (/ means all Objects)

Table 6: Bootstrap Discover parameters

For example:

- when the “Discover” operation targets an Object with Object ID of 0 (Security Object), the response to the operation could be:

  lwm2m="1.0",</0/0>;ssid=101, </0/1/>, </0/2>;ssid=102 with the meaning 3 LwM2M Servers are supported in that Client (that supports the LwM2M Enabler in version 1.0), while the Instance ID:1 of the Security Object ID:0 contains the credentials for the LwM2M Bootstrap-Server.

- when the “Discover” operation targets an Object with ‘/’ the response to the operation could be

  lwm2m="1.0",</0/0>;ssid=101,</0/1>, </1/0/>;ssid=101, </3/0>,</5>,</4> with the meaning the Client is supporting (in LwM2M version 1.0) a Bootstrap-Server Account (/0/1), a Server Account (/0/0, /1/0) with a Short Server ID=101, A Device Object Instance (/3/0) and 2 other Objects which are not instantiated yet (Firmware Update /5, and Connectivity Monitory Object /4).

- When the “Discover” addresses a LwM2M Client supporting the LwM2M Enabler in release 2.0, and containing a configuration with an hypothetical LwM2M Object ID:55 in Version 1.9, with one Instance (/55/0)

  lwm2m="2.0",</0/0>,</0/1>;ssid=101, </1/0/>;ssid=101, </3/0>,</5>,</4>,</55>;ver=1.9, </55/0>

5.2.7.4 BOOTSTRAP 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 targeted Object Instance(s) or Resource(s).

When the “Write” operation targets an Object or an Object Instance, the LwM2M Client MUST ignore optional resources it does not support in the payload. If the LwM2M Client supports optional resources not present in the payload, it MUST NOT instantiate these optional resources.

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.

5.2.7.5 BOOTSTRAP DELETE

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

Only in Bootstrap Interface, the Delete operation MAY targets "/0" URI. In that case, the operation MUST delete all the existing Security 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.

According to the Appendix D.1 rule related to Mandatory Object Instance, the mandatory Object Instances (e.g., ID:3) are not affected by the DELETE operation.

The Delete operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	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).

5.3 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– possibly without any parameters. If the lifetime of a registration expires without receiving an update from the LwM2M Client:

- the LwM2M Server MUST remove the registration of that Client;
- the LwM2M Client MUST re-register ("Update" is not sufficient) to the LwM2M Server in order to be connected again, before initiating any further communication.

[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).

If the LwM2M Server or the LwM2M Client set a value to the Lifetime Resource of the Server Object Instance, this value becomes the new lifetime of the Registration session.

During "Register" or "Update" Operations, the parameter Lifetime – if present – MUST match the current value of the Mandatory Lifetime Resource of the LwM2M Server Object Instance.

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.

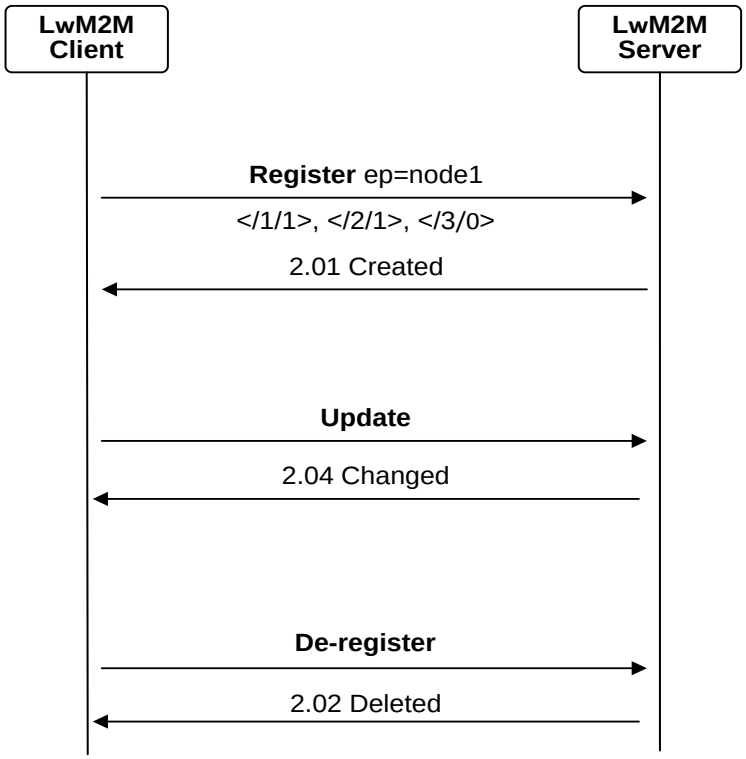


Figure 9: Client Registration Interface example flows

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 7 describes the parameters used for the "Register" operation.

The "Register" operation includes the Endpoint Client Name parameter along with other parameters listed in Table 7. 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 connection information of the registration message (e.g., source IP address and port or MSISDN) 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 7 and the LwM2M Client MUST support Endpoint Client Name, LwM2M Version, Lifetime, and Object and Object Instances and MAY support Binding Mode and SMS Number.

Parameter	Required	Default Value	Notes
Endpoint Client Name	Yes		See Section 0.6.3
Lifetime	Yes		Indicates the expected lifetime of the registration for this LwM2M client. This value MUST be the same as the value held in the Resource named "Lifetime" of the corresponding instance of Server Object (ID #1): /1/x/1.
LwM2M Version	Yes		Indicates the version of the LwM2M Enabler that the LwM2M Client supports.
Binding Mode	No	U	Indicates current binding and Queue mode of the LwM2M Client. This value MUST be the same as the value held in the Resource named "Binding" of the corresponding instance of Server Object (ID #1): /1/x/7. 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 (Security Object ID:0 MUST not be part of this list).

Table 7: Registration parameters

A LwM2M Server MUST refuse a Client's Registration request, if it doesn't support the LwM2M Enabler version indicated by the Client (i.e. major digit of the LwM2M versions in Client and Server are different, or the LwM2M version supported by the Client – e.g., 1.1 - is superior to the LwM2M version supported by the Server – e.g., 1.0 -).

The list of Objects and Object Instances is included in the payload of the registration message. Except the Security Object (ID:0), all the mandatory Objects defined in the LwM2M Enabler (i.e. Server Object ID:1 and the Device Object ID:3) MUST be part of the registration payload list. The Security Object ID:0 MUST NOT be part of the Registration Objects and Object Instances list.

When an Object defined outside of a LwM2M Enabler has to-be-registered by the Client, but is not supported by the Server (unknown Object or unsupported Object version) it MUST be silently ignored by this Server and will not prevent the Client's Registration request to be accepted.

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 augmented of the Object Version Core link parameters "ver" if required as it is defined in section 6.2 of this document. 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>,</2/3>,</2/4>,</3/0>,</4/0>,</5>

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 11543 is the value assigned in the CoAP Content-Format Registry for the LwM2M JSON format).

</>;ct=11543, </1/0>,</1/1>,</2/0>,</2/1>,</2/2>,</2/3>,</2/4>,</3/0>,</4/0>,</5>

5.3.1.1 Behaviour with Current Transport Binding and Mode

Behaviour 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 8 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 8: 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.

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. The LwM2M Client can perform an "Update" operation to refresh the lifetime of its registration to a LwM2M Server.

When any of the parameters listed in Table 9 changes, the LwM2M Client MUST send an "Update" operation to the LwM2M Server. This "Update" operation MUST contain only the parameters listed in Table 9 which have changed compared to the last registration parameters sent to the LwM2M Server.

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

Table 9: Update parameters

The Objects and Object Instance list MUST contain all the supported Objects and Object Instances available on the LwM2M Client. The Security Object ID:0 MUST NOT be part of Update Objects and Object Instances list.

When an Object defined outside of a LwM2M Enabler is part of the Client update registration list, but is not supported by the Server (unknown Object or unsupported Object version), it MUST be silently ignored by this Server and will not prevent the Client's Registration request to be accepted.

Two common operations are:

1) Extending the lifetime of a registration

In this case the Client sends a Registration Update with no parameters.

Figure 10 shows an example exchange where the Client sends a Registration Update message that only refreshes the registration, i.e., the message does not contain any parameters. With the second Registration Update the Client changes the lifetime field to 6000 (seconds) and hence the lt parameter is included in the message.

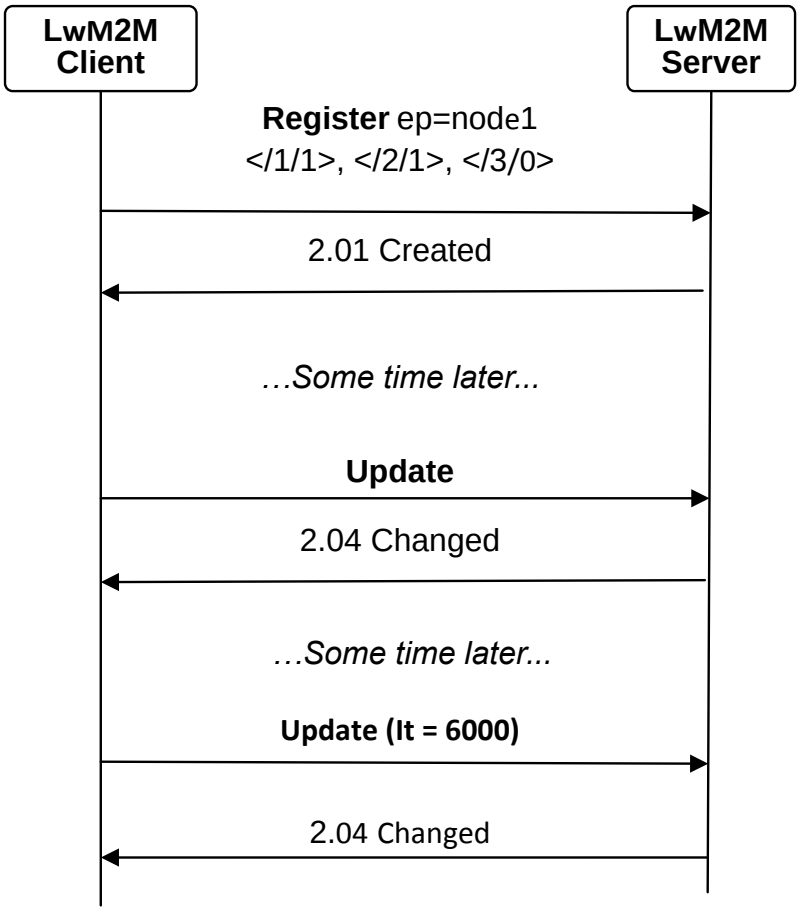


Figure 10: Client Registration Update example flows #1

2) Adding and removing Objects and Object Instances

In this case the client sends registration update with a body listing the complete list of objects and object instances.

Figure 11 shows an example exchange whereby the Client starts with an initial registration of two instances for a LwM2M Object with ID 12. Later, the server adds a third instance and subsequently deletes the second. In Registration Update messages, the Client includes the new list of Objects and Object Instances.

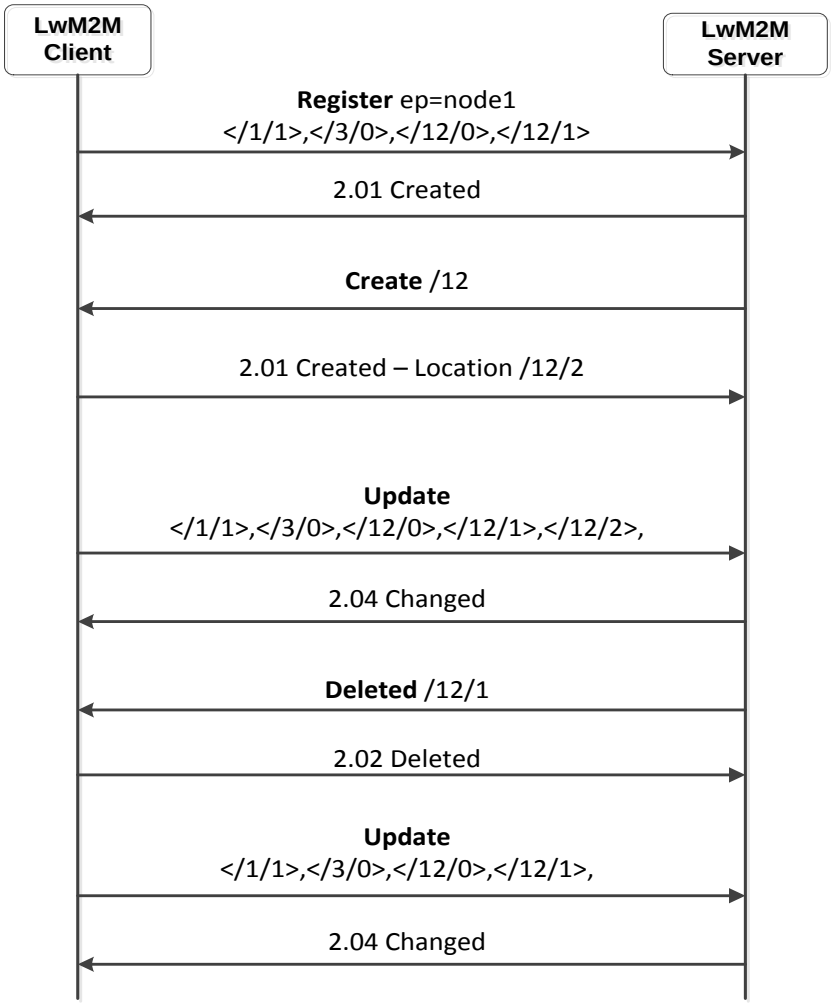


Figure 11: Client Registration Update example flows #2

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.

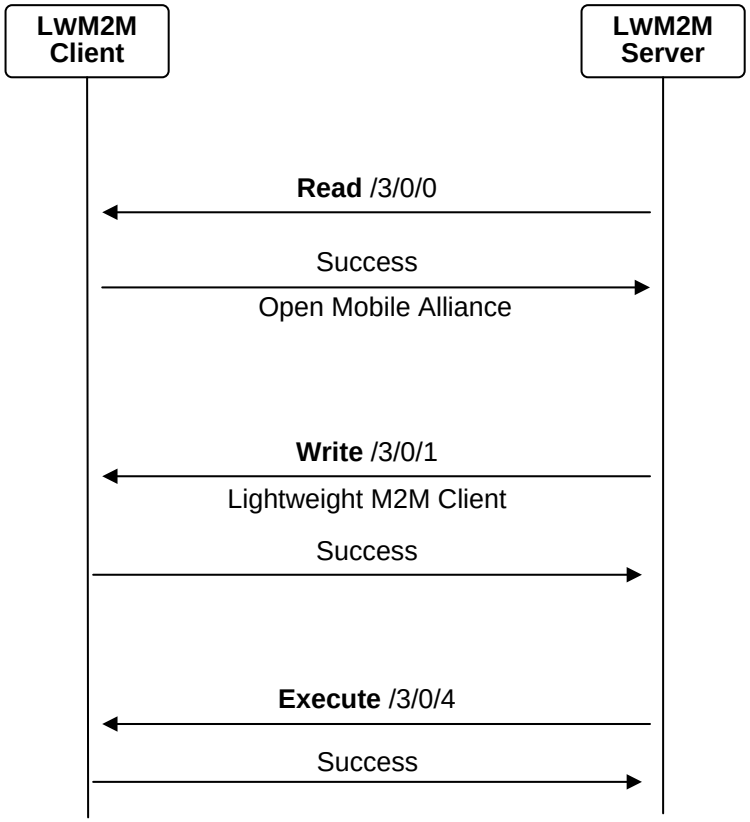
The Device Management and Service Enable Interface[2] is used by the LwM2M Server to access Object Instances and Resources available from a registered LwM2M Client. The interface provides this access through the use of "Create", "Read", "Write", "Delete", "Execute", "Write-Attributes", or "Discover" 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.

The LwM2M Client MUST ignore LwM2M Servers operations on this interface during a Server Initiated Bootstrap.

The LwM2M Client MUST ignore the LwM2M Server operation on this interface until it received its Registration acknowledgement.

[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).

The LwM2M Server SHOULD NOT perform any operation on this interface before replying to the registration of this LwM2M Client.



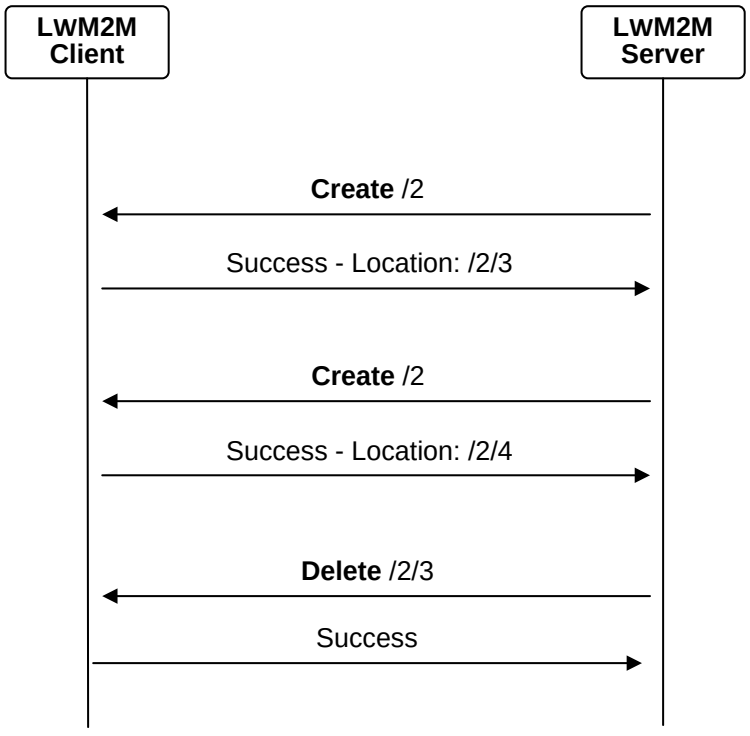


Figure 12: Example flows of Device Management & Service Enablement Interface

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 10: 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 instantiated in 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 including the Object Version attribute if required (see section 6.2 "Object Versioning" and Table 3 of this document <PROPERTIES> Class 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 11: 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 instantiated Resources. In addition, the list of Attributes which have be assigned to this Object level (see section 5.1.2 Attribute Classification) 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 supports the Device Info Object (Instance 0) Resources with IDs 1,2,3,4 6,7,8,11, and 16 among the Resources of Device Info Object, with an R-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 instantiated 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 supports the multiple Resources 6, 7, and 8 with a dimension of 8 and has 2 additional Notification parameters assigned for Resource 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 5.1.2 apply) For example: if Object ID is 3, and Resource ID is 7, then

</3/0/7>;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 the Attributes returned in the link, MUST only be related to the Server which performed the DISCOVER request. As example, for the Device Object (ID:3) having the Resource ID:7 with two Observe operations from two Servers 1 & 2, then the answers to DISCOVER command on both Servers will be differentiated e.g.,

  from Server 1: DISCOVER /3/0/7 could provide the answer: </3/0/7>;dim=8;gt=50;lt=42.2

  from Server 2: DISCOVER /3/0/7 could provide the answer: </3/0/7>;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 encoded in one of the data format defined in section 6.4.

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

The Write request MUST be rejected when the specified data 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 12: 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-Attributes" 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. When this parameter is omitted, the Resource ID parameter MUST be ignored and Attributes in the command parameters are valid for all resources of all instances of the Object according to the precedence rules (section 5.1.1).
Resource ID	No	-	Indicates the Resource. When this parameter is omitted, it means the Attributes of this commands are set at an upper level (Object or Object Instance level).
<NOTIFICATION> class Attributes	Yes		Indicates which Attributes are concerned. When an Attribute is specified without value, it means this Attribute value is unset at the level specified in the command (Object, Object Instance or Resource level).

Table 13: 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 14: 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 15: Create parameters

The new value included in the payload MUST follow the TLV or JSON data format according to section 6.4.

The LwM2M Client MUST ignore optional resources it does not support in the payload. If the LwM2M Client supports optional resources not present in the payload, it MUST NOT instantiate these optional resources.

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 16: 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 registered 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).

The LwM2M Client MUST ignore LwM2M Servers operations on this interface during a Server Initiated Bootstrap.

The LwM2M Client MUST ignore the LwM2M Server operation on this interface until it received its Registration acknowledgement.

The LwM2M Server SHOULD NOT perform any operation on this interface before replying to the registration of this LwM2M Client.

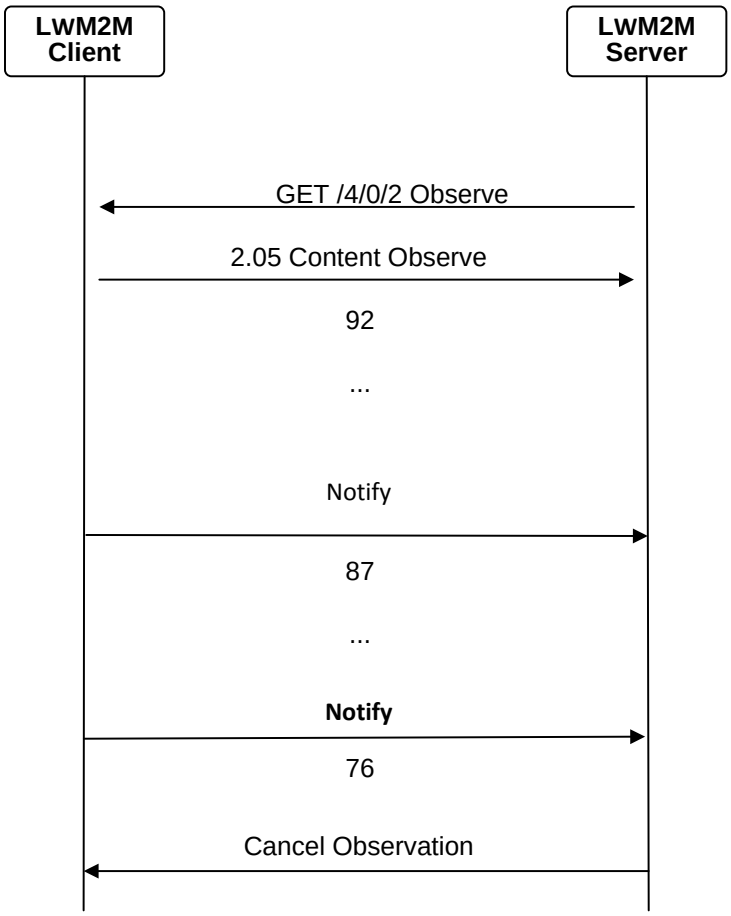


Figure 13: Example flow for Information Reporting Interface for the RSSI Resource of the Connectivity Monitoring Object of the example client (Appendix E)

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 17: Observe parameters

Until the LwM2M Client sends a new registration, the LwM2M Server expects the LwM2M Client to remember the observation requests. If a LwM2M Client forgets the observation requests (e.g., device factory reset) it MUST perform a new "Register" operation. The LwM2M Server MUST re-initiate observation requests whenever the LwM2M Client registers.

In order to avoid network traffic increase, the LwM2M Client SHOULD maintain previous observation states in case of reboot, power cycle.

It has to be noted that an "Observe" operation containing only Object ID, will produce the report of all Object Instances information.

Note: When a LwM2M Client deregisters, the LwM2M Server should assume past states are nullified including the previous observations.

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 18: 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.

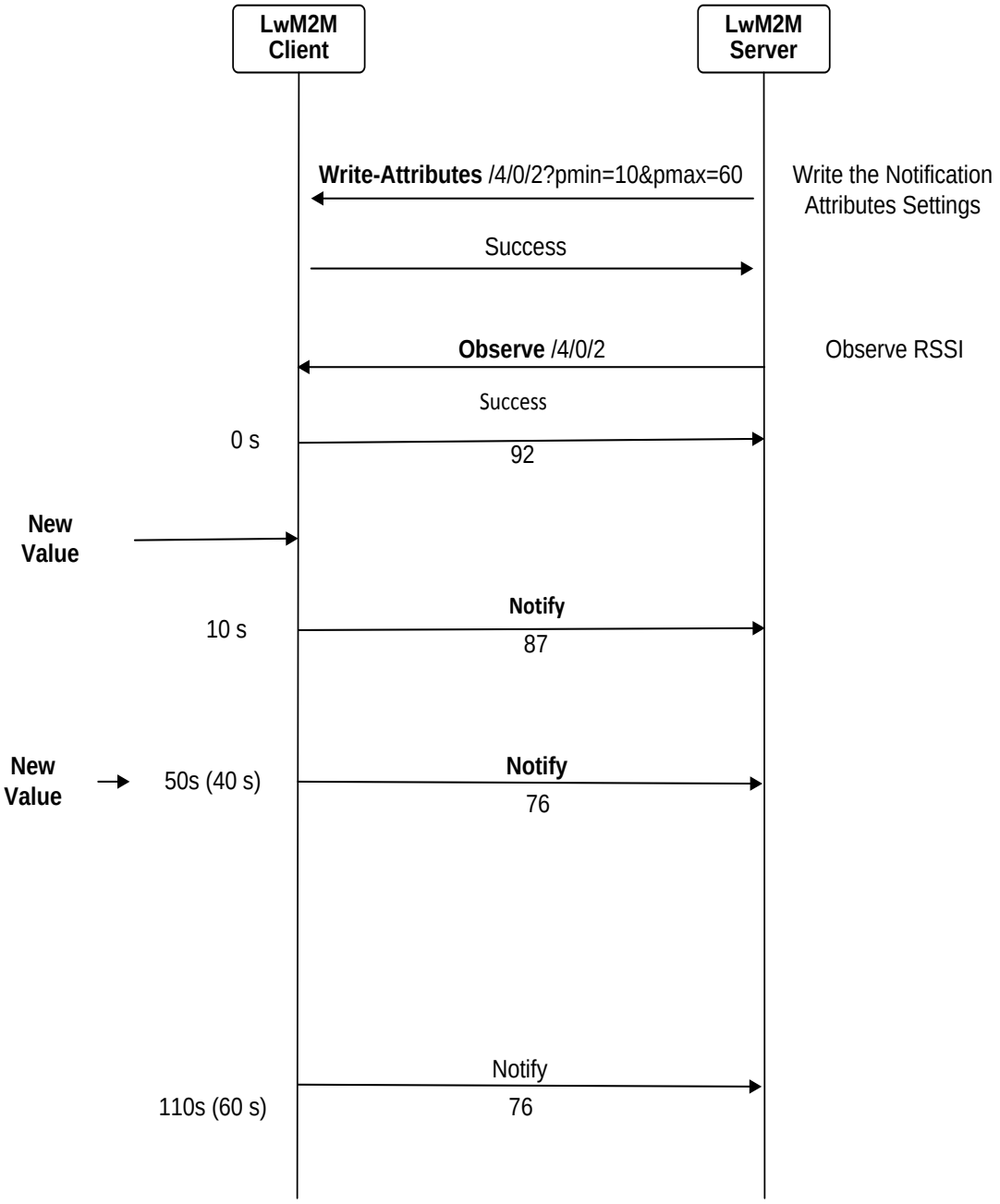


Figure 14: Example of Minimum and Maximum periods in an Observation

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.

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, and each Resource is given a unique identifier within that Object.

Figure 15 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; for example the Firmware Update Object contains all the Resources used for firmware update purposes.

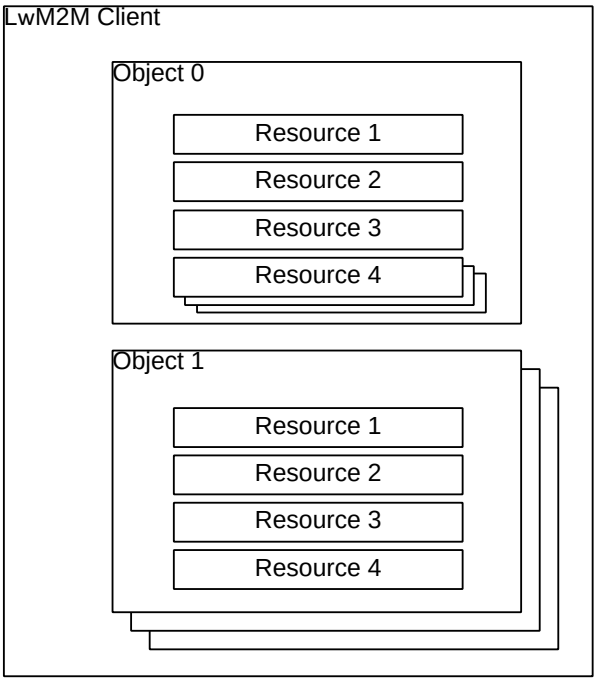


Figure 15: Relationship between LwM2M Client, Object, and Resources

Each Object, defined for the LwM2M Enabler, is assigned a unique OMA LwM2M Object identifier allocated and maintained by the OMA Naming Authority (OMNA). The LwM2M Enabler defines standard Objects and Resources (Appendix E). Further Objects may be added by OMA or other organizations to enable additional M2M Services.

As an Object only specifies a grouping of Resources, an Object MUST be firstly instantiated so that the LwM2M Client can use the Resources of such an Object and the associated functionalities.

When an Object is instantiated – either by the LwM2M Server or the LwM2M Client – an Object Instance is created with a subset of the Resources defined in the Object specification; a LwM2M Server can then access that Object Instance and its set of instantiated Resources.

A Resource which is instantiated within an Object Instance is a Resource which can either:

- contain a value (if the Resource is Readable and/or Writeable)
- or can be addressed by a LwM2M Server to trigger an action in the LwM2M Client (if the Resource is Executable)

The Object specification defines the operations (Read, Write, Execute) which are individually supported by the Resources belonging to that Object; this specification also defines the Mandatory or Optional characteristics of such Resources.

A Resource specified as Mandatory within an Object MUST be instantiated in any Instance of that Object.

A Resource specified as Optional within an Object MAY be omitted from some or even all Instances of that Object.

As illustration, the following example using the DISCOVER command on the Server Object, exposes a configuration in which the Server Object (ID:1) has 2 Instances (ID:0, ID:1): the Optional Resources ID:2, ID:4 are only instantiated in the Instance 1 of the Object, while the Optional Resources ID:3 and ID:5 are not instantiated in either of the Server Object Instances. In Server Object ID:1, the Resources 0,1, and 6,7,8 are Mandatory Resources.

According to the DISCOVER /1 request, the following payload is returned:

</1/0/0>,</1/0/1>,</1/0/6>,</1/0/7>, </1/0/8>, </1/1/0>,</1/1/1>,</1/1/2>,</1/1/4>,</1/1/6>,</1/1/7>, </1/1/8>

Objects and Resources have the capability to have multiple instances. Multiple-Instances Resources can be instantiated by LwM2M Server operations in using JSON or TLV formats (Section 5.4). The LwM2M Client also has the capability to instantiate Single or Multiple-Instances Resources.

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 8 Transport Layer Binding and Encoding and how to convey the Operation data is defined in 6.4.

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 Object Instance are allowed for which LwM2M Server(s).

Figure 16 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 1 has 2 Resources. Resource 1 supports the ”Read”, “Write” operations, while Resource 0 supports only the “Read” operation. The associated Access Control Object Instance has ACL of Object Instance 0 for Object 1. Server1 is authorized to perform “Read” and “Write” operations to the Object Instance 0 for Object 1 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 0, and also can perform the “Write” operations on Resource 1, but Server1 cannot perform the “Write” operation on Resource 0. The detailed access control mechanism is defined in Section 0.7.3 Access Control.

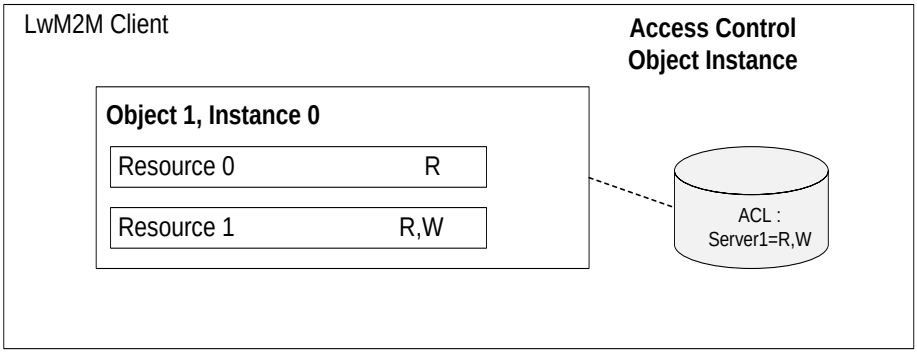


Figure 16: Example of Supported operations and Associated Access Control Object Instance

6.2 Object Versioning

6.2.1 General Policy

A LwM2M Object specification is tightly coupled with the LwM2M Enabler which is supporting it. However, a LwM2M Object that is not directly specified in the LwM2M Enabler but in a “companion specification”, MAY evolve (e.g., to fix one Object Resource characteristic issue).

For example:

- a Resource may be added or removed in the Object
- a Resource characteristic (mandatory nature, data type, operation mode …) may be changed in the Object

A LwM2M Server MUST be able to determine without ambiguity the specification of the Objects intended to be registered by the LwM2M Client:

- it is the case for the LwM2M Objects which are specified within the LwM2M

- for LwM2M Objects specified outside of a LwM2M Enabler, the Server has to be informed if the initial set of Resources characteristics have been modified/fixed, so that the Server can refer to the proper Object Specification

Object versioning aims at identifying the LwM2M Object modifications which can be categorized in 2 types defined below:

- Evolution of type I: representing non-backward-compatible ("breaking") evolutions, namely:
  - A Resource characteristic (optional vs mandatory, data type, supported operations) is changed in the Object
  - a Resource supporting a feature which is only defined by a new LwM2M Enabler, is added to the Object
- Evolution of type II: representing backward-compatible ("non-breaking") evolutions
  - an optional Resource is added or removed in the Object

In this document, the term "Initial Version" represents:

- the version 1.0 of the LwM2M Enabler
- the version 1.0 of any Object registered for the first time in OMNA

With Object versioning, the Object Identifier is not changed but a new URN identifying that particular version of the Object specification MUST be registered according to the following format

> "urn:oma:lwm2m:{oma, ext, x}:ObjectID[:{version}]" where 'version' is the Object Version as defined in the following section

6.2.2 Object Version format

The Object Version of an Object is composed of 2 digits separated by a dot '.':

- the first digit represents the Major Version of the Object: this digit marks the non-backward compatible evolution of such an Object (Object evolution of type I: section 6.2.1)
- the second digit represents the Minor Version of the Object: this digit marks the backward compatible evolution of such an Object (Object evolution of type II: see section 6.2.1)

By convention when an Object is in its Initial Version, the Object Version MAY be omitted in the URN.

To be more precise, two separate URNs are accepted as valid for the corresponding version of the object specification:

- the URN where the Object Version is mentioned explicitly: ″urn:oma:lwm2m:{oma, ext, x}:ObjectID:1.0″
- the URN where the Object Version is absent: ″urn:oma:lwm2m:{oma, ext, x}:ObjectID″ (and supposed to be 1.0)).

Example: "urn:oma:lwm2m:oma:44:2.2"

This URN uniquely identifies Object ID:44 in its version "2.2". Registered Objects are available on the OMNA portal (see Section 6.2.3 and the Appendix J on Object Definition File for further information).

6.2.3 Object Definition and Object Version Usage

The rules governing in which circumstances the Object Version is provided by the LwM2M Client to the LwM2M Server during the "Register" and "Discover" operations, are synthesized in the following table:

	Object in Initial Version (1.0)	Object Version > 1.0
Objects defined within LwM2M TS Enabler	The LwM2M Client MAY provide the Object Version	The LwM2M Client MAY provide the Object Version
Objects defined Outside any LwM2M TS Enabler	The LwM2M Client MAY provide the Object Version	The LwM2M Client MUST provide the Object Version

Table 19: Object Version usage rules

In “Register” and “Discover” operations, when the Object Version of a given Object must be communicated, the LwM2M Client MUST use the “ver” (object version) <PROPERTIES> Class attribute associated to that Object.

Few examples:

a) URN: “urn:oma:lwm2m:oma:44:2.2” Object 44 in version 2.2

Details:

- on the OMNA portal the Object ID:44 will be registered at least twice
  - with the URN “urn:oma:lwm2m:oma:44” (Initial Version)
  - with the URN “urn:oma:lwm2m:oma:44:2.2” (or the latest release of the major Version 2, LwM2M Server ignores minor releases)
- the Object definition contains the minimal version of the LwM2M Enabler supporting that Object (which can be omitted if this LwM2M version is the “Initial Version” one)
- LwM2M Client “Register” operation: ep=nodename,</1/0>,</1/1>,</3/0>,</44>;ver= “2.2”,</44/0>

b) URN: “urn:oma:lwm2m:oma:3”

Details: LwM2M Client “Register” operation: ep=nodename,</1/0>,</1/1>,</3/0>

c) URN: “urn:oma:lwm2m:oma:44”

Details: LwM2M Client “Register” operation: ep=nodename,</1/0>,</1/1>,</3/0>,</44/0>

d) URN: “urn:oma:lwm2m:oma:3” in LwM2M Client supporting LwM2M Enabler version 2.0

Details:

- the minimal version of the LwM2M Enabler supporting that Object “3” (Device) is still LwM2M 1.0; this information may be omitted in the Device (ID:3) Object Definition file
- LwM2M Client “Register” operation: lwm2m=“2.0”,ep=nodename,</1/0>,</1/1>,</3/0>

e) URN: “urn:oma:lwm2m:oma:4:3.1”

Context:

- the LwM2M Client supports LwM2M Enabler version 2.0
- Object ID: 4 supports a specific feature of LwM2M 2.0 not supported by the LwM2M Enabler Initial Version, and the Object Version 3 is part of the LwM2M Enabler Version 2.0

Details:

- the minimal version of the LwM2M Enabler supporting that Object “4” (Device) must be indicated in the Object 4 version 3.1 Definition file: LwM2MVersion=“2.0” and ObjectVersion=“3.1”
- LwM2M Client “Register” operation: lwm2m=“2.0”,ep=nodename,</1/0>,</1/1>,</3/0>,</4/0>

6.3 Identifiers

The LwM2M Enabler defines specific identifiers for entities used within the LwM2M Protocol. These identifiers are defined in Table 20.

Identifier	Semantics	Description

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 6.3.1 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 integer	Uniquely identifies the Object Instance of 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 20: LwM2M Identifiers

6.3.1 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:##############
IMEI-MSISDN URN: Identify a device using a combination of International Mobile Equipment Identifier [3GPP-TS_23.003] and MSISDN. IMEI is 15 digits and MSISDN is 15 digits. The format of the URN is urn:imei-msisdn: ###############-###############

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

6.3.2 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.

6.4 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. In addition a LwM2M Client MAY choose to support other data formats in the table below i.e., JSON, plain text and opaque.

Messages containing data MUST specify the payload encoding by using one of the supported data format.

A LwM2M Server data request MAY contain an option specifying the data formats the Server would prefer to receive for the payload; if this data format is not accepted by the LwM2M Client, the request is rejected; if the LwM2M Client doesn't support that option or if the LwM2M Server expresses no data format preference, the LwM2M Client will use its own preferred data format.

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	11542
JSON	application/vnd.oma.lwm2m+json	11543

6.4.1 Plain Text

The plain text format is used for "Read" and "Write" operations on singular Resources where the value of the Resource is represented as described in 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

6.4.2 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.

6.4.3 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 21: 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.

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

For simplicity and to prevent any ambiguity on Object Instance Identity, when the Object Instance ID is not specified in the request, the Object Instance TLV MUST be used, whatever the number of Object Instances (1 or more) to return to the LwM2M Server.

When a Multiple Resource has to be returned to the LwM2M Server, the Multiple Resource TLV MUST be used whatever the number of Instances (0, 1 or more) of that resource.

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.

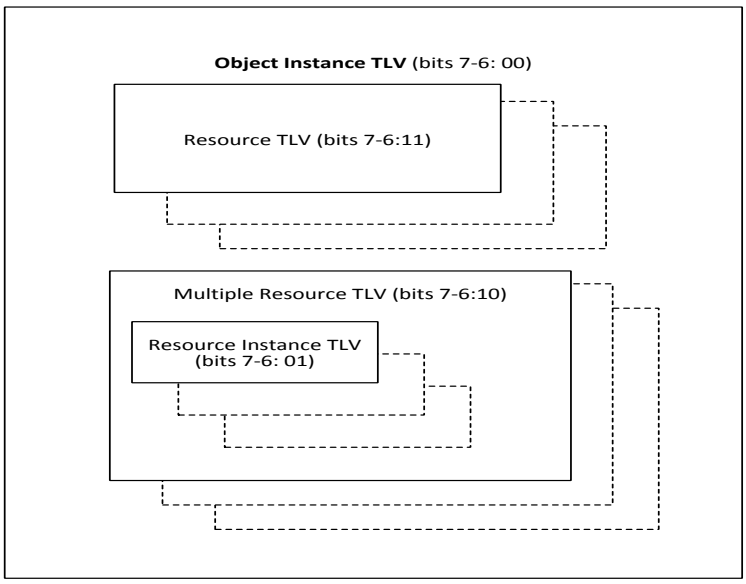


Figure 17: TLV nesting

6.4.3.1 Single Object Instance Request Example

In this example, a request for the Device Object Instance (ID:3) of a LwM2M client is made (Read /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:

```
C8 00 14 4F 70 65 6E 20 4D 6F 62 69 6C 65 20 41 6C 6C 69 61 6E 63 65
C8 01 16 4C 69 67 68 74 77 65 69 67 74 20 4D 32 4D 20 43 6C 69 65 6E 74
C8 02 09 33 34 35 30 30 30 31 32 33
C3 03 31 2E 30
86 06
   41 00 01
   41 01 05
88 07 08
   42 00 0E D8
   42 01 13 88
87 08
   41 00 7D
   42 01 03 84
C1 09 64
C1 0A 0F
83 0B
   41 00 00
C4 0D 51 82 42 8F
C6 0E 2B 30 32 3A 30 30
C1 10 55
```

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	–	"1.0" [String] (3 bytes)	5
Available Power Sources	0b10 0 00 110	0x06	–	The next two rows (6 bytes)	2
Available Power Sources[0]	0b01 0 00 001	0x00	–	0X01 [8-bit Integer]	3
Available Power Sources[1]	0b01 0 00 001	0x01	–	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	–	0X0ED8 [16-bit Integer]	4
Power Source Voltage[1]	0b01 0 00 010	0x01	–	0X1388 [16-bit Integer]	4
Power Source Current	0b10 0 00 111	0x08	–	The next two rows (7 bytes)	2
Power Source Current[0]	0b01 0 00 001	0x00	–	0X7D [8-bit Integer]	3
Power Source Current[1]	0b01 0 00 010	0x01	–	0X0384 [16-bit Integer]	4
Battery Level	0b11 0 00 001	0x09	–	0x64 [8-bit Integer]	3
Memory Free	0b11 0 00 001	0x0A	–	0x0F [8-bit Integer]	3
Error Code	0b10 0 00 011	0x0B	–	The next row (3 bytes)	2
Error Code[0]	0b01 0 00 001	0x00	–	0x00 [8-bit Integer]	3
Current Time	0b11 0 00 100	0x0D	–	0x5182428F [32-bit Integer]	6
UTC Offset	0b11 0 00 110	0x0E	–	"+02:00" [String] (6 bytes)	8
Supported Binding and Modes	0b11 0 00 001	0x10	–	"U" [String] (1 byte)	3
				Total	121

6.4.3.2 Multiple Object Instance Request Examples

A) Request on Single-Instance Object

In this example, a request for the Device Object Instance (ID:3) of a LwM2M client is made (Read /3). The Device Object is a Single-Instance Object, but the Object Instance TLV is used (to be compared with the example of the previous section).

The client responds with a TLV payload including the Object Instance ID and all its readable Resources. This TLV payload would have the following value (indented for readability) and format. The total payload size with the TLV encoding is 124 bytes:

```
08 00 79
   C8 00 14 4F 70 65 6E 20 4D 6F 62 69 6C 65 20 41 6C 6C 69 61 6E 63 65
```

```
C8 01 16 4C 69 67 68 74 77 65 69 67 74 20 4D 32 4D 20 43 6C 69 65 6E 74
C8 02 09 33 34 35 30 30 30 31 32 33
C3 03 31 2E 30
86 06
   41 00 01
   41 01 05
88 07 08
   42 00 0E D8
   42 01 13 88
87 08
   41 00 7D
   42 01 03 84
C1 09 64
C1 0A 0F
83 0B
   41 00 00
C4 0D 51 82 42 8F
C6 0E 2B 30 32 3A 30 30
C1 10 55
```

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Device Object Instance 0	**0b00 0 01 000**	**0x00**	**0x79 (121 bytes)**	**The next 20 rows**	**3**
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	—	“1.0” [String] (3 bytes)	5
Available Power Sources	0b10 0 00 110	0x06	—	The next two rows (6 bytes)	2
Available Power Sources[0]	0b01 0 00 001	0x00	—	0X01 [8-bit Integer]	3
Available Power Sources[1]	0b01 0 00 001	0x01	—	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	—	0X0ED8 [16-bit Integer]	4
Power Source Voltage[1]	0b01 0 00 010	0x01	—	0X1388 [16-bit Integer]	4
Power Source Current	0b10 0 00 111	0x08	—	The next two rows (7 bytes)	2

Power Source Current[0]	0b01 0 00 001	0x00	—	0X7D [8-bit Integer]	3
Power Source Current[1]	0b01 0 00 010	0x01	—	0X0384 [16-bit Integer]	4
Battery Level	0b11 0 00 001	0x09	—	0x64 [8-bit Integer]	3
Memory Free	0b11 0 00 001	0x0A	—	0x0F [8-bit Integer]	3
Error Code	0b10 0 00 011	0x0B	—	The next row (3 bytes)	2
Error Code[0]	0b01 0 00 001	0x00	—	0x00 [8-bit Integer]	3
Current Time	0b11 0 00 100	0x0D	—	0x5182428F [32-bit Integer]	6
UTC Offset	0b11 0 00 110	0x0E	—	"+02:00" [String] (6 bytes)	8
Supported Binding and Modes	0b11 0 00 001	0x10	—	"U" [String] (1 byte)	3
				Total	124

B) Request on Multiple-Instances Object having 2 instances

In this example, a request on the Access Control Object (ID:2) of a LwM2M client is made (Read /2). The Access Control Object is a Multiple-Instances Object. In this simplified example, it has only 2 instances describing the access rights of two LwM2M Servers with Short IDs 127 and 310 on Objects Instances /1/0 and /3/0.

The client responds with a TLV payload including the 2 Object Instances (ID:0 and ID:2) and their Resources. This TLV payload would have the following value (indented for readability) and format. The total payload size with the TLV encoding is 38 bytes:

```
08 00 0E
   C1 00 01
   C1 01 00
   83 02
      41 7F 07
   C1 03 7F
08 02 12
   C1 00 03
   C1 01 00
   86 02 41 7F 07 61 01 36 01
   C1 03 7F
```

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Access Control Object Instance 0	**0b00 0 01 000**	**0x00**	**0x0E (17 bytes)**	**The next 5 rows**	**3**
Object ID	0b11 0 00 001	0x00	—	0x01 [8-bit Integer]	3
Object Instance ID	0b11 0 00 001	0x01	—	0x00 [8-bit Integer]	3
ACL	0b10 0 00 011	0x02	—	The next row (3 bytes)	2
ACL [127]	*0b01 0 00 001*	*0x7F*	—	*0b000 00111* [8-bit Integer]	3

Access Control Owner	0b11 0 00 001	0x03	–	0x7F [8-bit Integer]	3
Access Control Object Instance 2	**0b00 0 01 000**	**0x02**	**0x12 (18 bytes)**	**The next 6 rows**	**3**
Object ID	0b11 0 00 001	0x00	–	0x03 [8-bit Integer]	3
Object Instance ID	0b11 0 00 001	0x01	–	0x00 [8-bit Integer]	3
ACL	0b10 0 00 110	0x02	–	The next 2 rows	2
ACL [127]	*0b01 0 00 001*	*0x7F*	–	*0b000 00111* [8-bit Integer]	3
ACL [310]	*0b01 1 00 001*	*0x0136*	–	*0b000 00001* [8-bit Integer]	4
Access Control Owner	0b11 0 00 001	0x03	–	0x7F [8-bit Integer]	3
				Total	38

C) Request on Multiple-Instances Object having 1 instance only

In this example, a request to the Server Object Instances of a LwM2M client is performed (Read /1). The Server Object is a Multiple-Instances Object. The client has only one Server Object instance and will respond with a TLV payload including the single Object Instance (0) and their Resources. This TLV payload would have the following value (indented for readability) and format. The total payload size with the TLV encoding is 18 bytes:

```
08 00 0D
   C1 00 01
   C4 01 00 01 51 80
   C1 06 01
   C1 07 55
```

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Server Object Instance 0	**0b00 0 01 000**	**0x00**	**0x0D (13 Bytes)**	**The next 5 rows**	**3**
Short Server ID	0b11 0 00 001	0x00	–	0x01 [8-bit Integer]	2
Lifetime	0b11 0 00 100	0x01	–	86400 [32-bit Integer]	5
Notification Storing When Disabled or Offline	0b11 0 00 001	0x06	–	True [Boolean]	3
Binding	0b11 0 00 001	0x07	–	"U" [String] (1 byte)	3
				Total	16

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

Examples are based on the LwM2M Object Tree illustration of Figure 28. The TLV format doesn't report Object hierarchy.

Example 1) request to Object 65 Instance 0: Read /65/0

```
88 00 0C
   44 00 00 42 00 00
```

```
   44 01 00 42 00 01
C8 01 0D 38 36 31 33 38 30 30 37 35 35 35 30 30
C4 02 12 34 56 78
```

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Res 0 lnk	0b1 0 0 01000	0x00	0x0C(12 bytes)	The next 2 rows	3
Res 0 lnk [0]	0b01 000 100	0x00	—	0x0042 0000 [Objlnk] (66:0)	6
Res 0 lnk [1]	0b01 0 00 100	0x01	—	0x0042 0001 [Objlnk] (66:1)	6
Res 1	0b11 0 01 000	0x01	0x0D	“8613800755500” [String] (13 bytes)	16
Res 2	0b11 0 00 100	0x02	—	0x12345678 [32-bit Integer]	6
				Total	37

Example 2) request to Object 66: Read /66: TLV payload will contain 2 Object Instances

```
08 00 23
        C8 00 0B 6D 79 53 65 72 76 69 63 65 20 31
        C8 01 0F 49 6E 74 65 72 6E 65 74 2E 31 35 2E 32 33 34
        C4 02 00 43 00 00
08 01 23
        C8 00 0B 6D 79 53 65 72 76 69 63 65 20 32
        C8 01 0F 49 6E 74 65 72 6E 65 74 2E 31 35 2E 32 33 35
        C4 02 FF FF FF FF
```

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Object 66 Instance 0	**0b00 0 01 000**	**0x00**	**0x23 (35 bytes)**	**The next 3 rows**	3
Res 0	**0b11 0 01 000**	0x00	0x0B	“myService 1” [String] (11 bytes)	14
Res 1	**0b11 0 01 000**	0x01	0x0F	“Internet.15.234” [String] (15 bytes)	15
Res 2 lnk	0b11 0 00 100	*0x02*	—	0x0043 0000 [Objlnk] (67:0)	6
Object 66 Instance 1	**0b00 0 01000**	**0x01**	**0x23 (35 bytes)**	**The next 3 rows**	**3**
Res 0	0b11 0 01 000	0x00	0x0B	“myService 2” [String] (11 bytes)	14
Res 1	0b11 0 00 000	0x01	0x0F	“Internet.15.235” [String] (15 bytes)	15
Res 2 lnk	0b11 0 00 100	0x02	—	0xFFFF FFFF [Objlnk] (no link)	6
				Total	76

6.4.4 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 22.

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 attribute need to be prepended by the Base Name attribute to form the unique identifier of this Resource instance.

The Name attribute in of this array entry is a URI path relative to the Base Name; namely the Name attribute could simply be the full URI of a requested Resource Instance 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.

Note: According to [SENML] specification, time values (Base Time and Time) are represented in floating point as seconds, a missing attribute is considered to have a value of zero.

Regarding the Base Time, "*Positive values (>0) represent an absolute time relative to the unix epoch, while negative values (<= 0) represent a relative time in the past from the current time*" [SENML].

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 time which the Time values are relative to.
Resource Array	e		Yes	The Resource list as JSON value array according to [SENML] with Array parameter extension (Object Link).
	Array Parameters			
	Name	n	No	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 Array entry Name is the full URI of the requested Resource Instance

	Time	t	No	The time of the representation relative to the Base Time in seconds 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 22: JSON format and description

For example a request to Device Object of the LwM2M example client (Read /3/0) would return the following JSON payload. This example has a size of 457 bytes.

```
{"bn":"/3/0/",
 "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":"16","sv":"U"}]
}
```

For example a notification about a Resource containing multiple historical representations of a Temperature Resource (ID:2) (e.g. Instance ID:1 of hypothetical Object ID 72) could result in the following JSON payload:

```
{"bn":"/72/",
  "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 65 of the LwM2M example from Figure 28 (Read /65/0) would return the following JSON payload.

Because the Base Name is specified, the full hierarchy linked to the Instance 0 of Object 65 can be reported in a single response (Object 66 Instance 0 & 1, and Instance 0 of Object 67 are part of the payload). This example has a size of 435 bytes.

```
{ "bn":"/",
  "e":[
  {"n":"65/0/0/0","ov":"66:0"},
```

```
   {"n":"65/0/0/1","ov":"66:1"},
   {"n":"65/0/1","sv":"8613800755500"},
   {"n":"65/0/2","v":1},
   {"n":"66/0/0","sv":"myService1"},
  {"n":"66/0/1","sv":"Internet.15.234"},
   {"n":"66/0/2","ov":"67:0"},
   {"n":"66/1/0","sv":"myService2"},
  {"n":"66/1/1","sv":"Internet.15.235"},
   {"n":"66/1/2","ov":"FFFF:FFFF"},
   {"n":"67/0/0","sv":"85.76.76.84"},
   {"n":"67/0/1","sv":"85.76.255.255"}]
}
```

For example, a request to Device Object on Resource 0 of the LwM2M example client (Read /3/0/0) could return the following JSON payload.

```
{"bn":"/3/0/0",
 "e":[
   {"sv":"Open Mobile Alliance}]
}
```

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 DTLS with these channel bindings to implement authentication, confidentiality, and data integrity features of the protocol between communicating LwM2M entities. As an alternative, lower layer security may be used, as described in Section 7.2.

LwM2M Clients require credentials and configuration information for securely communicate with LwM2M Servers. This configuration information can be provisioned to the LwM2M Client during manufacturing or through the use of the LwM2M Bootstrap-Server. In order to secure the communication between the Lwm2M Client and the LwM2M Bootstrap-Server a different set of credentials and configuration information is required.

LwM2M supports three different types of credentials, namely

- Certificates,
- Raw public keys, and
- Pre-shared secrets.

Since these credential types offer different properties the LwM2M offers support for all of them.

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 DTLS-based Security

7.1.1 Requirements

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.

Due the sensitive nature of bootstrap information, a particular care has to be taken to ensure protection of that data.

The use of DTLS fulfils these requirements.

7.1.2 DTLS Overview

CoAP [CoAP] is secured using the Datagram Transport Layer Security (DTLS) 1.2 protocol [RFC6347], which is based on TLS v1.2 [RFC5246]. The DTLS binding for CoAP is defined in Section 9 of [CoAP]. DTLS is a communication security solution for datagram based protocols (such as UDP). It provides a secure handshake with session key generation, mutual authentication, data integrity and confidentiality.

This section provides information related to the use of DTLS for use with CoAP over DTLS over UDP as well as for use with CoAP over DTLS over SMS. Section 7.3 provides additional information regarding the use of DTLS in an SMS context.

The DTLS client and the DTLS server SHOULD keep security state, such as session keys, sequence numbers, and initialization vectors, and other security parameters, established with DTLS for as long a period as can be safely achieved without risking compromise to the security context. If such state persists across sleep cycles where the RAM is powered off, secure storage SHOULD be used for the security context.

The credentials used for authenticating the DTLS client and the DTLS server to secure the communication between the LwM2M Client and the LwM2M Server are obtained using one of the bootstrap modes defined in Section 5.2.2. Appendix E.1.1 defines the format of the keying material stored in the LwM2M Security Object Instances.

LwM2M Bootstrap-Servers, LwM2M Servers and LwM2M Clients MUST use different key pairs. LwM2M Clients MUST use keys, which are unique to each LwM2M Client. When a LwM2M Client is configured to utilize multiple LwM2M Servers then the LwM2M Bootstrap-Server may configure different credentials with these LwM2Ms Servers. Such configuration provides better unlinkability properties since each individual LwM2M Server cannot correlate request based on the credentials used by the LwM2M Client. Deployment and application specific considerations dictate what approach to use.

7.1.3 Ciphersuites

DTLS supports the concept of ciphersuites and they are securely negotiated during the DTLS handshake. This specification recommends a limited number of ciphersuites. The recommended ciphersuites have been chosen because of suitability for IoT devices, security reasons and to improve interoperability and depend on the type of credential being used since the ciphersuite concept also indicates the authentication and key exchange mechanism. LwM2M Clients and LwM2M Servers MAY support additional ciphersuites that conform to state-of-the-art security requirements.

Note that care has to be taken when using CBC-based ciphersuites in DTLS for the following two reasons:

(1) Prior to TLS 1.1 IV selection is broken. The solution is to use TLS 1.1 or higher, and there is a work-around for earlier version using record splitting. Since this specification relies on DTLS 1.2 this concern is not applicable.

(2) Implementing authenticated decryption (checking padding and mac) without any side channel is hard (see Lucky 13 attack and its variants). The solution is to use the encrypt-then-mac extension defined in RFC 7366, which is recommended.

7.1.4 Bootstrapping

The Resources in the LwM2M Security Object (i.e., “Security Mode”, “Public Key or Identity”, “Server Public Key or Identity” and “Secret Key”) are used

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.

3) for protecting the communication with a firmware repository server when the LwM2M Client receives a URI in the Package URI of the Firmware Update object.

The content and the interpretation of the Resources in the LwM2M Security Object depend on the type of credential being used.

Concerning Bootstrap from Smartcard a secure channel between the Smartcard and the LwM2M Client SHOULD be established, as described in Appendix G and defined in [GLOBALPLATFORM 3], [GP SCP03]. Using Smartcard with pre-shared secrets, raw public keys, and with certificates needs no pre-existing trust relationship between LwM2M Server(s) and LwM2M Client(s). The pre-established trust relationship is between the LwM2M Server(s) and the SmartCard(s).

LwM2M Clients MUST either be provisioned for use with a LwM2M Server (manufacturer pre-configuration bootstrap mode) or else be provisioned for use 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 Section 7.1.7. The ciphersuites defined in Section 7.1.7 MUST NOT be used with a low-entropy secret or with a password.

2) Bootstrapping with a raw public key or certificate-based method (as described in Section 7.1.8 and Section 7.1.9).

In either case, the LwM2M Client MUST be provisioned with a credential that is unique to a device. For full interoperability, a LwM2M Bootstrap-Server MUST support bootstrapping via pre-shared secrets, raw public keys, and certificates.

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).

Security credential dynamically provisioned to the LwM2M Client and the LwM2M Server MAY change at any time, even during the lifetime of an ongoing DTLS session. Since the DTLS protocol verifies the credentials only at the beginning of the session establishment (unless the re-negotiation feature is used) it is possible that a change in credential (for example, credentials for the use of a PSK-based ciphersuite) occurs after a DTLS handshake has already been completed and the DTLS session setup is already finalized. Hence, from a DTLS protocol point of view such a change is not recognized and the already established record layer security associations are in use. It is a policy decision for a DTLS client as well as a DTLS server implementation to tear down an already existing session when the credentials change. Such a decision will depend on various factors, such as the application domain in which LwM2M is used. The LwM2M specification does not mandate a specific behaviour in such a case since DTLS allows both communication parties to tear down an established DTLS session for any number of reasons.

7.1.5 Endpoint Client Name

The LwM2M specification defines the use of the endpoint client name in the Bootstrap-Request and in the Register messages. Since the endpoint client name is not authenticated at the application layer the LwM2M Server MUST compare the received endpoint client name identifier with the identifier used at the DTLS handshake. This comparison may either be an equality match or may involve a dedicated lookup table to ensure that LwM2M Clients cannot intentionally or due to misconfiguration impersonate other LwM2M Clients. The LwM2M Server MUST respond with a “4.00 Bad Request” to the LwM2M Client if these fields do not match.

7.1.6 LwM2M and DTLS Roles

The client-server roles of DTLS, which indicate who initiates the DTLS handshake, are independent from the client-server relationship of LwM2M. In client-initiated bootstrapping the LwM2M Client is also the DTLS client and the LwM2M Bootstrap Server acts as the DTLS server. For server-initiated bootstrapping, however, the roles are reversed: the LwM2M Client acts in the role of a DTLS server and the LwM2M Bootstrap Server is the DTLS client. Note that using a DTLS server on a LwM2M Client requires additional resources, such as RAM, and flash memory.

When the LwM2M Client acts in the role of a DTLS server then care has to be taken that the following four values are equal:

(1) Value in the Server Name Indication (SNI) extension used in the DTLS exchange,

(2) Endpoint Client Name,

(3) Identifier used with the credential, such as the identifier contained in the DTLS server certificate, and

(4) Value in the LwM2M Server URI Resource.

Note that the DTLS client (acting as the LwM2M Server) for the server-initiated bootstrapping has to be configured with the IP address of the LwM2M Client, an FQDN, and the certificate, raw public key or PSK for use with the LwM2M Client.

7.1.7 Pre-Shared Keys

A LwM2M Server MUST support the Pre-Shared Key mode of DTLS with the following ciphersuites:

- TLS_PSK_WITH_AES_128_CCM_8, as defined in [RFC6655]
- 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 ciphersuites specified for the LwM2M Server.

This mode requires the following resources of the Security Object defined in Appendix E.1 to be populated:

- The “Security Mode” Resource MUST contain the value 0.
- The "Public Key or Identity" Resource MUST be used to store the PSK identity, defined in [RFC4279].
- The "Secret Key" Resource MUST be used to store the PSK, defined in [RFC4279].
- The “Server Public Key” Resource MUST NOT be used in the Pre-Shared Key mode.

7.1.8 Raw Public Keys

If a LwM2M Server supports the raw public key credentials it MUST support the following ciphersuites:

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, as defined in [RFC6655]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, as defined in [RFC5289]

If a LwM2M Client supports the raw public key mode it MUST support at least one of the ciphersuites supported by the LwM2M Server.

This mode requires the following resources of the Security Object defined in Appendix E.1 to be populated:

- The “Security Mode” Resource MUST contain the value 1.
- The "Public Key or Identity" Resource MUST be used to store the raw public key of the DTLS client.
- The "Secret Key" Resource MUST be used to store the private key of the DTLS client.
- The “Server Public Key” Resource MUST be used to store the raw public key of the DTLS server.

This security mode is appropriate for LwM2M deployments where the benefits of asymmetric cryptography are used but without the overhead of the public key infrastructure.

The DTLS client MUST check that the raw public key presented by the DTLS server exactly matches this stored public key.

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

7.1.9 X.509 Certificates

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

If a LwM2M Server supports X.509 Certificate mode it MUST support the following ciphersuites:

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, as defined in [RFC7251].
- 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 ciphersuites supported by the LwM2M Server.

This mode requires the following resources of the Security Object defined in Appendix E.1 to be populated:

- The “Security Mode” Resource MUST contain the value 2.
- The "Public Key or Identity" Resource MUST be used to store the X.509 certificate of the DTLS client.
- The "Secret Key" Resource MUST be used to store the private key of the DTLS client.
- The “Server Public Key” Resource MUST be used to store the certificate of the DTLS server. The use of it is explained in more detail below.

The "LwM2M Server URI", and the "Bootstrap Server" Resources are populated according to the description in Appendix E.1.

The public key infrastructure supports different deployment modes, as discussed in [RFC6698], and this specification supports the domain issued certificate mode whereby the Server Public Key Resource specifies the exact certificate that should be used for the DTLS server, and the certificate does not need to be signed by a valid CA. This allows for the use of self-signed certificates. Other modes are not supported.

The algorithm for verifying the service identity, as described in RFC 6125 [RFC6125], is essential for ensuring proper security when certificates are used and MUST be implemented and used by the DTLS client. Terms like reference identifier and presented identifier are defined in RFC 6125.
Comparing the reference identifier against the presented identifier obtained from the certificate is required to ensure the DTLS client is communicating with the intended DTLS server. Since only the domain-issued certificate mode is supported by this specification the DTLS client compares the certificate from the Server Public Key Resource with the certificate provided

in the DTLS handshake additionally comparing the reference identifier against the presented identifier is step for future-proofing in the anticipation of supporting other PKIX validation modes. Similarly, a DTLS client running on a LwM2M Server would need to obtain the certificate of the DTLS server running on the LwM2M Client from some repository.
The algorithm description from RFC 6125 assumes that fully qualified DNS domain names are used. If a server node is provisioned with a fully qualified DNS domain, then the DTLS server certificate MUST contain the fully qualified DNS domain name or "FQDN" as dNSName [RFC5280]. For CoAP, the coaps URI scheme is described in Section 6.2 of [RFC7252]. This FQDN is stored in the SubjectAltName or in the leftmost Common Name (CN) component of the subject name, as explained in Section 9.1.3.3 of [RFC7252], and used by the DTLS client to match it against the FQDN used during the lookup process, as described in [RFC6125].
Note that the Server Name Indication (SNI) extension [RFC6066] allows a DTLS client to tell a DTLS server the name of the DTLS server it is contacting. This is an important feature when the server is part of a hosting solution where multiple virtual servers are using a single underlying network address. Section 3 of [RFC6066] only allows FQDN hostname of the DTLS server in the ServerName field. For the DTLS client running on a LwM2M Server the SNI extension allows the LwM2M Server to indicate what certificate it is expecting.
In some deployment scenarios DNS is not used and hence LwM2M Clients need to follow a different procedure.
If the CoAP URI stored in the "LwM2M Server URI" Resource contains an IP literal, such as coaps://[2001:db8::2:1]/, then certificate provided by the server also has to contain such an IP address in the Common Name (CN) component of the server certificate or in a field of URI type in the SubjectAltName set.
The procedure for a client using such certificates is as follows:

- The LwM2M Client uses the IP address from the LwM2M Server URI Resource to connect to the LwM2M Server using a DTLS handshake. The IP address becomes the reference identifier.
- The DTLS stack of the LwM2M Server returns a Certificate message as part of the handshake that contains a certificate. The IP address extracted from the server certificate becomes the presented identifier.
- The client matches the reference identifier against the presented identifier. If the two match, the client continues with the certificate verification according to RFC 5280 and aborts the handshake with a fatal alert otherwise.

There are disadvantages in the way how IP addresses are used in the LwM2M specification with certificates. Whenever the IP address of the LwM2M Server changes a new certificate for that LwM2M Server needs to be created. Due to the only supported domain-issued certificate mode the LwM2M Server certificate also needs to be provisioned to the LwM2M Client since otherwise the DTLS handshake will fail since the certificate provisioned to the Server Public Key Resource will not match the newly generated LwM2M Server certificated provided in the DTLS handshake. Furthermore, the IP address contained in the LwM2M Server URI Resource will also need to be updated. Finally, IP addresses cannot be used in the SNI extension.

The use of certificates requires the DTLS client to understand the concept of time since it needs to check the validity of the server-provided certificate. Different deployments may have different means of obtaining the current time and this specification does not mandate one mechanism. In general, the LwM2M Bootstrap-Server certificate is not expected to expire since the LwM2M Client has no easy possibility to recover from such an expired certificate. However, if the LwM2M Client determines that the LwM2M Server certificate is expired it MAY contact the LwM2M Bootstrap-Server to obtain new security credentials for use with the LwM2M Server.

Note that the LwM2M Device Object allows the LwM2M Bootstrap-Server to configure the current time for the LwM2M Client using the Current Time Resource.

7.1.10 “NoSec” mode

It is highly recommended to always protect the LwM2M protocol with DTLS. There are, however, scenarios where the LwM2M protocol is deployed in environments where lower layer security mechanisms are provided.

The LwM2M Server MUST compare the endpoint client name identifier used during the Register and the Bootstrap-Request message with the identifier used for network access authentication (typically used to setup link layer security procedures).

The LwM2M protocol may use the NoSec mode with or without a lower-layer security mechanism and matching the endpoint client name identifier with any lower layer identifier may in the latter case not be possible.

7.1.11 Certificate mode with EST

This mode uses the configuration of the certificate mode defined in Section 7.1.9 with the following changes; instead of generating the certificate and the private key for the client by the LwM2M Bootstrap Server and to provision it to the LwM2M Client the Bootstrap Server MUST set the “Security Mode” Resource to value 4 and provisions the certificate of the DTLS server to the “Server Public Key” Resource. This triggers the LwM2M Client to locally generate a public / private key

pair on the LwM2M Client and to initiate an EST over CoAP protocol exchange [CoAP-EST] to obtain a certificate. The EST over CoAP specification [CoAP-EST] profiles the use of EST for use in constrained environments.

When generating a public / private key pair, the random generator used by the LwM2M Client MUST respect the characteristics of a sufficiently high quality random bit generator, such as defined for example by ISO/IEC 18031:2011, RFC 4086 [RFC4086] or NIST Special Publication 800-90a [SP800-90A].

Compared to the certificate mode additional over-the-air overhead is introduced by this mode since the LwM2M Client needs to convey the public key to the EST server and needs to demonstrate possession of the private key using the PKCS#10 defined mechanism, as referenced in the EST specification. Depending on the deployment environment this additional overhead needs to be compared against the added security benefit of not disclosing the private key to other parties.

The "Secret Key" and the "Public Key or Identity" Resources are not used by this mode. The "LwM2M Server URI", and the "Bootstrap Server" Resources are populated according to the description in Appendix E.1.

Enrolment over Secure Transport (EST) offers multiple features, including

- Simple PKI messages,
- CA certificate retrieval,
- CSR Attributes Request,
- Server-generated key request,

but only the first two are mandatory to implement.

In context of this specification functionality for server-generated key requests is already covered as part of the security mode (1 - Raw Public Key mode and 2 - Certificate mode). CSR Attributes Request is also not required for this specification either since the LwM2M Bootstrap Server is typically in possession of the required attributes for generating a certificate. The CA certificate retrieval, while mandatory to implement for EST, is not used by version 1.0 of this specification since only the domain issued certificate mode is supported, as described in Section 7.1.9. Hence, CA certificates are not utilized.

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 recommended.

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 "DTLS-based Security" relative to DTLS binding on CoAP are also applicable to that section.

The following sub-sections 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. The text has been re-used from Appendix A of [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

According to the recommendation of [SMS-DTLS], the SMS Channel media in LwM2M requires to follow the two rules below:

- the 3GPP Concatenated Short Message mechanisms MUST NOT be used
- the same PMTU setting used during the DTLS Handshake 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 would require to subtract additional 7 bytes (WDP header) from the SMS effective payload and is not supported in LwM2M release 1.0. Later version of OMA LwM2M may support it through a new SMS DTLS mode (DTLS 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

To deal with the unreliable message delivery provided by UDP, DTLS adds timeouts and "per-flight" retransmissions, as described in Section 4.2.4 of [RFC6347]. Although the timeout values are implementation specific, recommendations are provided in Section 4.2.4.1 of [RFC6347], with an initial timer value of 1 second and double the value at each retransmission, up to no less than 60 seconds.

An initial timer value of 9 seconds with exponential back off up to no less than 60 seconds is therefore RECOMMENDED.

This value is chosen big enough to absorb large latency variance due to intrinsic network characteristics, as well as to produce a low number of retransmission events and relax the pacing between them.

Its worst case wait time is the same as using 1s timeout (i.e., 63s), while triggering less than half of the retransmissions (2 instead of 5).

In order to minimize the wake time during DTLS handshake, sleepy nodes might decide to select a lower threshold and, consequently, a smaller initial timeout value. If this is the case, the implementation MUST keep into account the considerations about network stability described in this section.

When the SMS delivery report function is activated, the reception of an SMS-STATUS-REPORT message has not to be interpreted as an indication that a previously sent handshake message has been acted on by the receiver. Therefore, the SMS-STATUS-REPORT message MUST NOT be considered by the DTLS timeout and retransmission function. In order to avoid caching messages in the network, the SMS validity period carried by the handshake messages MUST have a value higher or 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="5">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 requires 29 bytes: 13 bytes header according to (RFC 6347 and Appendix B of [RFC7925] + 8 bytes for the explicit nonce and 8 bytes for the integrity check value when an AES-128-CCM-8 ciphersuite is used. This ciphersuite uses a short integrity check value.
    - CoAP header of variable length with at least 4 bytes [CoAP]
    - Available bytes for the effective LwM2M payload from one SMS: 107 bytes

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

<table>
<tr><td colspan="5">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 + 8 bytes)</td><td>Effective Payload (99 bytes)</td></tr>
</table>

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 address

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 Appendix 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 certain Object or 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, the Access Control Object MAY also be instantiated in a single LwM2M2 Server context. In the case the Access Control Object is instantiated in a single LwM2M2 Server context, section 7.3.1 and its sub-sections MUST also be applied that context.

7.3.1 Access Control Object

7.3.1.1 Access Control Object overview

In the presence of several LwM2M Servers, there is a need to determine if a certain LwM2M Server is authorized to instantiate a supported Object in the LwM2M Client. This kind of authorization can only be managed during a Bootstrap Phase.

Furthermore, the LwM2M Client needs to determine - per supported Object Instance - who the "Access Control Owner" of that Object Instance is and which access rights have been granted to other LwM2M Servers likely to interact with that LwM2M Client on such Object Instance.

The Access Control Object is specified in Appendix E.3 and Examples of Access Control Object Instances are presented in Appendix F.

7.3.1.2 Access Control Object Management

7.3.1.2.1 Access Control on Object

An Access Control Object Instance MUST be associated per Object supported by the LwM2M Client, to specify which LwM2M Server is authorized to instantiate ("Create" operation) such an Object.

- This kind of Access Control Object Instance associated with a certain Object, MUST only be created or updated during a Bootstrap Phase. If such association doesn't exist, an Access Control Object Instance MUST be created with the Resources values which MUST be set as given in the table just below.

- when such Access Control Object Instance already exists for a certain Object, this Access Control Object Instance MAY be updated for supporting additional LwM2M Servers; in that case a new ACL Instance per Server is created in that Access Control Object Instance with the Short Server ID of the LwM2M Server as index of this new ACL Instance, and the "C" access right as ACL Resource value.

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 5th LSB: "Create" is only configured
Access Control Owner	3	MAX_ID=65535 (meaning: managed by Bootstrap Interface)

7.3.1.2.2 Access Control on Object Instance

An Access Control Object Instance MUST be associated per Object Instance supported by the LwM2M Client, to register which operations MAY be performed by a certain LwM2M Server on this associated Object Instance.

This kind of Access Control Object Instance associated with a certain Object Instance, MAY be created or updated either during a Bootstrap Phase or through the Device Management and Service Enablement Interface.

In particular:

- when a LwM2M Server creates under authorization an Object Instance (see section 7.3.2 Authorization) in the LwM2M Client, an Access Control Object Instance MUST be created in the LwM2M Client with the Resources values which MUST be set as given in the table just below. The Access Control Owner Resource is configured with the 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 combination of the Access Right {none,R,W,E,D} is acceptable (Appendix E.3)

Access Control Owner	3	The Short Server ID of the LwM2M Server owner of the associated Object Instance

- when this Access Control Object Instance is created during the Bootstrap Phase, ACL(s) and Access Control Owner MUST be set with values respecting the consistency of the LwM2M Client configuration.
- through the Device Management and Service Enablement Interface, an Access Control Object Instance MUST only be managed by the LwM2M Server declared as the "Access Control Owner" in it.
- when the LwM2M Server - which is the "Access Control Owner" - adds or modifies (using "Write" operation) access right on the Object Instance for a certain LwM2M Server,
    a. 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. 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
- A specific ACL Resource Instance MAY be used to grant access rights to LwM2M Servers (except the one defined as the Access Control Owner) which don't have their own ACL Resource Instance. The ID of this ACL Resource Instance containing the default access rights MUST be 0.
- 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.

The Figure 18 illustrates the Access Control Management of an Object Instance supported by a LwM2M Client, in using an Access Control Object Instance: the path 2 locates the Access Control Object Instance in charge of the targeted Object Instance addressed by the path 1. ACL instances in 3, refer to the LwM2M Servers and the access rights granted to them by the LwM2M Server owner of the targeted Object Instance.

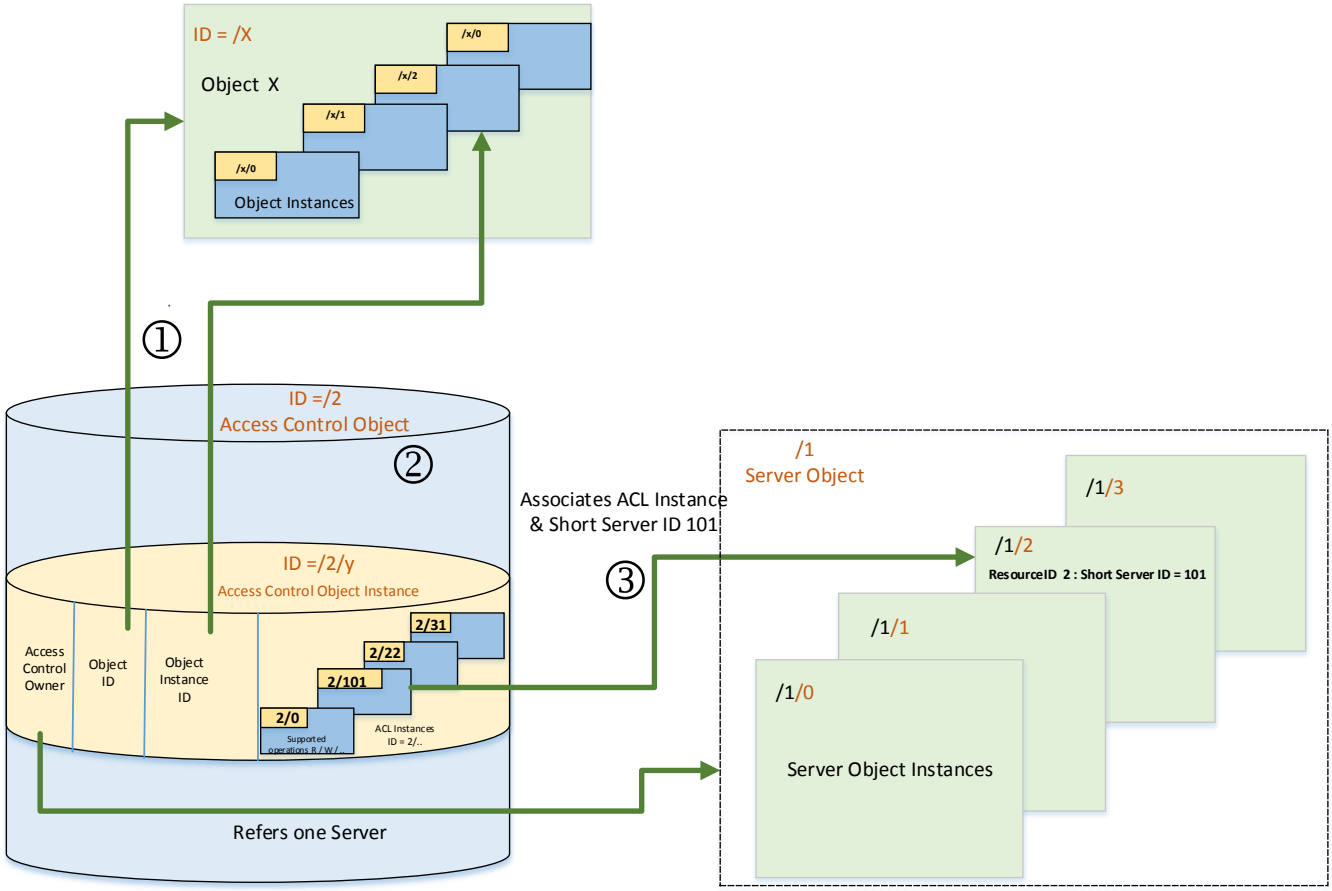


Figure 18: Illustration of the relations between the LwM2M Access Control Object and the other LwM2M Objects

7.3.1.3 LwM2M Server Context Switch

In a single LwM2M Server context, Access Control Object MAY NOT be instantiated. In that case, adding a new LwM2M Server Account, means this initial single Server context MUST be switched to a multi-Server context in which Access Control Object MUST be instantiated. This context switch must be managed in a Bootstrap Phase.

Note: The Bootstrap Discover command allows to retrieve data regarding the configuration of the initial single LwM2M Server context (list of Objects, Object Instances and Short Server IDs), the Bootstrap Server has then all information to setup a proper multi-Server context from this initial single Server context one.

7.3.2 Authorization

The LwM2M Client MUST authorize a CREATE operation requested by a LwM2M Server for instantiating a certain Object, only if the associated Access Control Object Instance, contains an ACL Resource Instance for that LwM2M Server set with the Access Right “Create” (section 7.3.1.2.1).

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 LwM2M Server is authorized to perform the operation, the LwM2M Client still needs to check if the LwM2M Server requested operation is supported by the targeted Resource or Object Instance (details are provided in section 7.3.2.2, 7.3.2.3, and 0.7.37.3.27.3.2.4).

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 (i.e. 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 ACL Resource Instance (ID:0) containing the default access rights if it exists (Section 7.3.1.2.2).

If the Client doesn't have this ACL Resource Instance ID:0 containing the default access rights, 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 check if 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 if 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 perform 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 if 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 that the Object Instance cannot perform the requested "Write" operation in 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 if 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 present in the "New Value" parameter (see Section 5). If all the mandatory Resources are not present, 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; if a value of such a Read-only Resource is present in the "New Value" parameter, this value MUST simply be ignored. If the payload (New Value) 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 if 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-Format Option MUST be used to indicate the data format 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.
- CoAP Blockwise transfer for CoAP MUST be supported by the LwM2M Client when the Firmware Update Object (ID:5) is implemented by the client and MUST be supported by the LwM2M Server.
  This functionality is motivated by limitations of CoAP, as defined in RFC 7252 [CoAP] since CoAP was not designed for transmission of large payloads. Because the CoAP header itself does not contain length information the UDP length header is used instead. The maximum UDP datagram size is limited to ~64 KiB and transmitting data beyond the (path) maximum transmission (MTU) size will additionally lead to inefficiency because of fragmentation at lower layers (IP layer, adaptation layer, and link layer). Blockwise Transfer for CoAP [draft-ietf-core-block-20] was specifically designed to lift this limitation in order to transfer large payloads larger than ~64 KiB via CoAP, such as firmware images.
  Note: [CoAP_Blockwise] is also beneficial for use with firmware images smaller than 64 KiB since the block-wise transmission allows the server to deliver firmware images in chunks suitable to the MTU and thereby avoiding fragmentation at lower layers. A LwM2M client may choose to support block-wise transfer for objects other than the Firmware Update object. This may, for example, be useful with objects that are larger in size, such as the security object which may contain certificates. The specifics of how this functionality is utilized by a LwM2M Server are out of scope for this release of LwM2M.

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 Alternate Path

By default, the LwM2M Objects are located under the root path. However, devices might be hosting other CoAP Resources on an endpoint, and there may be the need to place LwM2M Objects under an alternate path.

When registering, or updating its registration, a LwM2M Client MAY include an OMA LwM2M link in addition to the Object links in the registration payload. The link is identified by the RFC6690 Resource Type parameter "oma.lwm2m".

This link MUST NOT contain numerical URI segment.

For instance, the Example Client from Appendix F may place Objects under the "/lwm2m" path. The registration payload would be as follows:

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

When using the Device Management & Service Enablement Interface and the Information Reporting Interface, the LwM2M Server MUST prepend the OMA LwM2M link to the path in the CoAP messages. Example: GET /lwm2m/3/0/0.

When using the Bootstrap Interface, the LwM2M Bootstrap-Server MUST use CoAP paths only in the form /{Object ID}/{Object Instance ID}/{Resource ID}. It is the responsibility of the LwM2M Client to map these paths to its alternate path.

The Resource Type value "oma.lwm2m" is part of IANA registry.

8.2.3 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.

During the Bootstrap Phase, the Client MAY ignore requests and flush all pending responses not related to the Bootstrap sequence.

In client initiated bootstrap, when the Bootstrap-Server receives Bootstrap-Request operation, or in server initiated bootstrap, the Bootstrap-Server performs Write, Discover and/or Delete operations. The Delete operation targets an Object or an Object Instance, the Discover operation targets an Object, while a Write operation targets Object, Object Instance or a Resource. The Write, Discover and Delete operations 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.

Only in Bootstrap Interface, the Discover command MAY target to "/" URI to discover all Objects and Object Instances supported in the Device.

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
Discover	GET Accept: application/link-format	/{Object ID}	2.05 Content	4.00 Bad Request 4.04 Not Found
Bootstrap-Finish	POST	/bs	2.04 Changed	4.00 Bad Request 4.06 Not Acceptable

Table 23: Operation to Method and URI Mapping

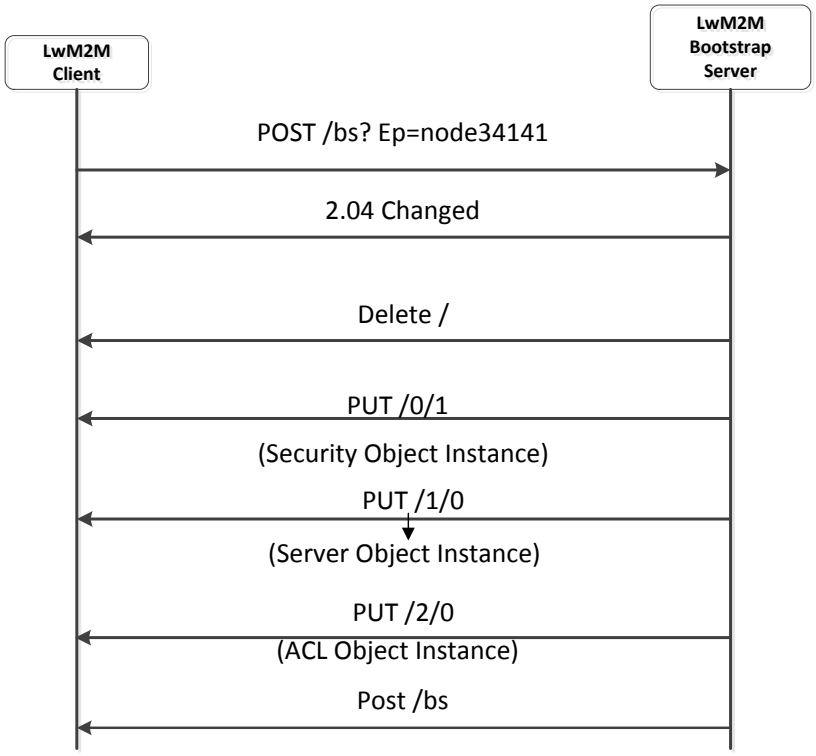


Figure 19: Example of Client initiated Bootstrap exchange

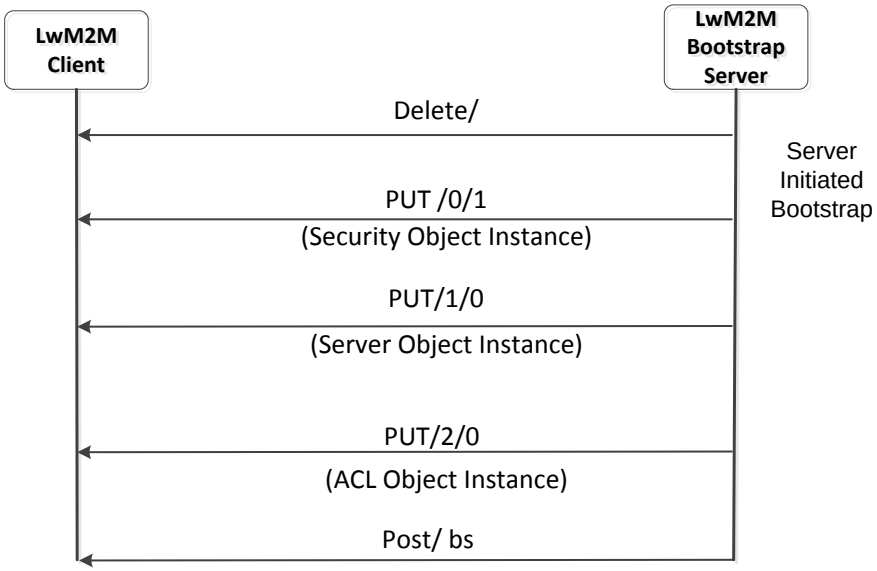


Figure 20: Example of Server initiated Bootstrap exchange

8.2.4 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 /rd, with registration parameters passed as query string parameters as per Table 24 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 LwM2M Server MUST return a location under the /rd path segment.

As the network connectivity may be limited or intermittent, it is advised to make several retries of the Registration if no reply is received from the LwM2M Server before considering the registration as failed.

When a new DTLS Session is started, or in NoSec mode when the LwM2M Client IP address changes, the Client MUST register again to the LwM2M Server.

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, 4.12 Precondition Failed
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 24: Operation to Method and URI Mapping

Note: Throughout the present document the format of the MSISDN must be as specified in [3GPP-TS_23.003]. According to this definition "+" is not preceding the country code.

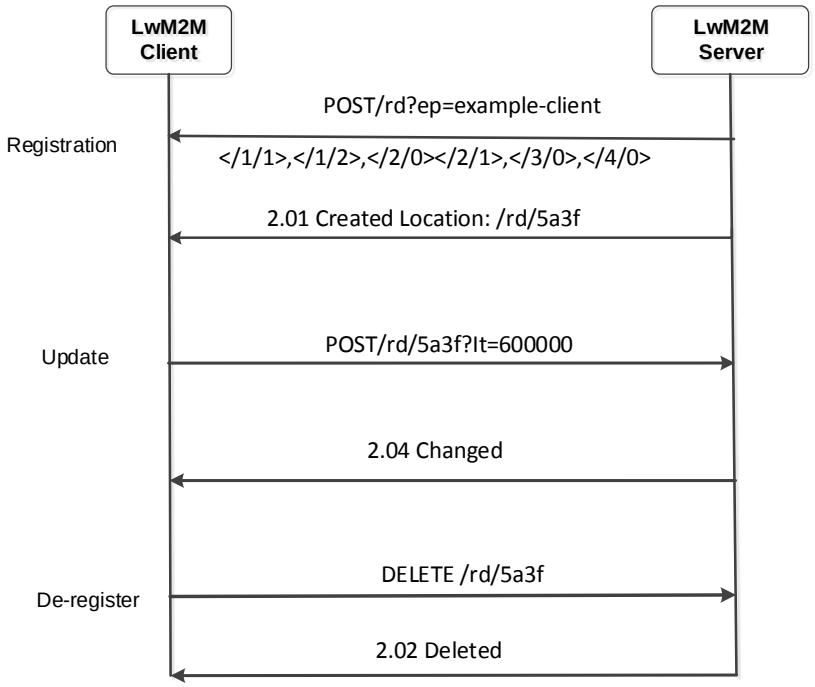


Figure 21: Example register, update and de-register operation exchanges (shorthand in [CoAP] example style, actual messages using CoAP binary headers)

8.2.5 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 6.4).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 (5.2.4).

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 by sending a CoAP PUT on the corresponding path, and can be accessed using the "Discover" operation. The Discover operation (uses a CoAP GET on the corresponding path along with the application/link-format Content type, to retrieve a list of Objects, Object Instances, Resources and their attached attributes, from the LwM2M Client (see Section 5.4.2 for more details on DISCOVER command). With the "Write-Attributes" 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 set to which Attribute and at which level (Object / Object Instance / Resource). In a similar way, the same command without value for the specified Attribute, MUST be used to unset 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; this value is set at the Resource level.

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 set at another level (Object or Object Instance: see section 5.1.1).

c) Write-Attributes /3/0?pmin=10 means that all Resources of Instance 0 of the Object 'Device (ID:3)' will be notified to the Server with a minimum interval of 10 sec; this value is set at the Object Instance level.

Operation	CoAP Method	Path	Success	Failure
Read	GET Accept: Content Format ID (see section 6.4)	/{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}	2.31* Continue	4.08* Request Entity Incomplete 4.13* Request entity too large
Write-Attributes	PUT	/{Object ID}/{Object Instance ID}/{Resource ID}?pmin={minimum period}&pmax={maximum period}&gt={greater than}&lt={less than}&st={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}	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

Note (*): 2.31, 4.08, 4.13 response messages, are relevant only when the CoAP Blockwise Transfer option is supported (see Section 8.1).

Table 25: Operation to Method Mapping

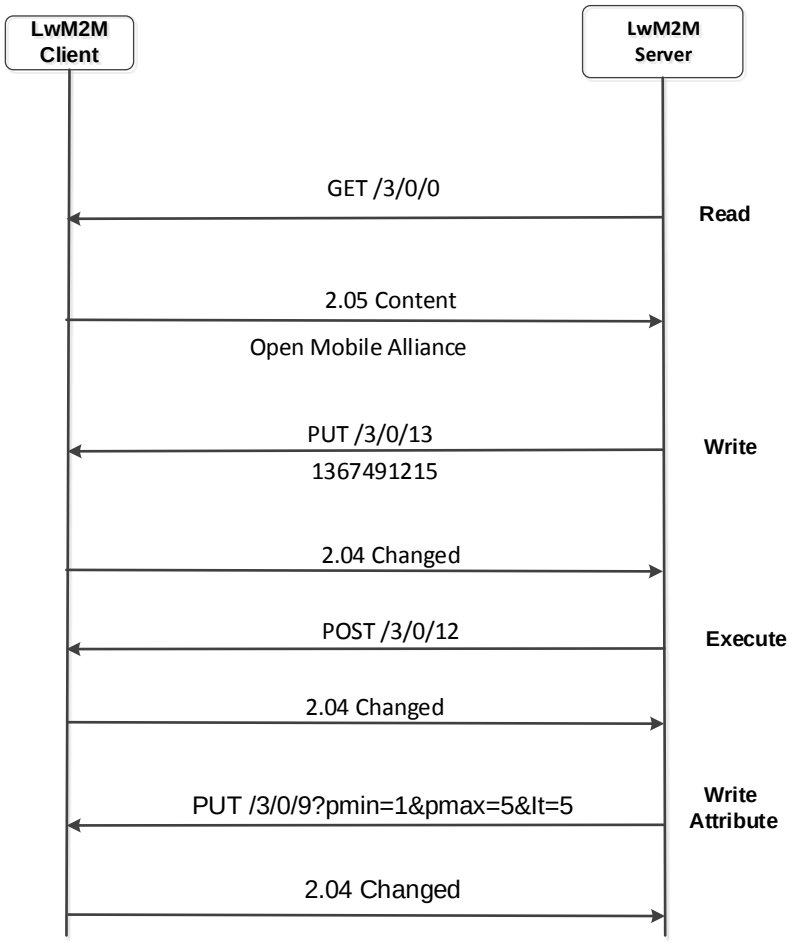


Figure 22: Example of Device Management & Service Enablement interface exchanges

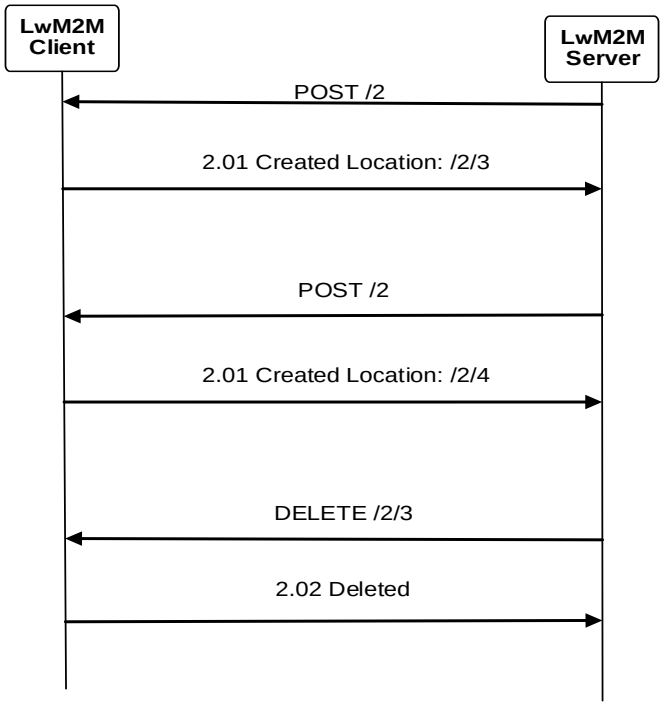


Figure 23: Example of Object Creation and Deletion

8.2.6 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).

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).

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 26: Operation to Method Mapping

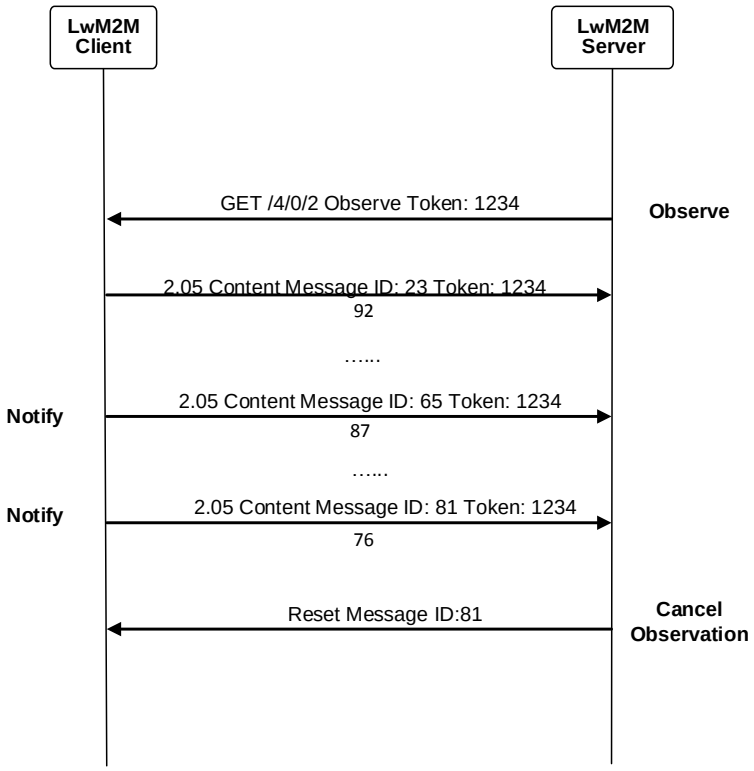


Figure 24: Example of an Information Reporting exchange

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 used in Queue Mode, but instead waits until the LwM2M Client is online. As such, the Queue Mode offers functionality for a LwM2M Client to inform the LwM2M Server that it may be disconnected for an extended period of time and also when it becomes reachable again. The LwM2M Server uses this information to adjust timers and relay messages from and to the LwM2M Client accordingly.

The LwM2M Client lets the LwM2M Server know it is awake by sending a registration update message as a Confirmable message. Absent any application specific profiles it is RECOMMENDED that the LwM2M Client waits at least MAX_TRANSMIT_WAIT seconds [CoAP] from the last CoAP message it sent to the LwM2M Server before intentionally going offline.

In order to find out whether a message was successfully delivered from the LwM2M server to the LwM2M client the LwM2M server has to rely on a response. This response tells the server that the message has been received and processed (regardless of what the result of the processing was). A response can be conveyed to the server in two ways:

- ACK piggybacking the response, or
- Separate CON/non-CON containing the response.

If message delivery fails, for example, because the message got lost due to network connectivity issues or because the LMW2M Client was sleeping then CoAP re-transmission behaviour at the LwM2M Server will try to retransmit the message. The CoAP stack at the LwM2M Server will resend the message up to a certain number of attempts, as described in Section

4.2 of [CoAP]. If these retransmission attempts fail, the CoAP stack at the LwM2M Server will give up and inform the LwM2M layer. The LwM2M Server has to inform the application about this failed delivery but this API is outside the scope of the LwM2M specification.

Due to the congestion control approach used by CoAP the LwM2M Server has to wait for a response to a request before sending out the next request from the queue since [CoAP] limits the number of simultaneous outstanding interactions to 1.

Despite the title of the functionality, i.e. Queue Mode, this specification does not mandate an implementation to use queues nor does it specify where such a queue would exist (or any details of such queuing mechanism).

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 is recommended to use the CoAP MAX_TRANSMIT_WAIT parameter to set a timer for how long it shall stay awake since last sent message to the LwM2M Server. After MAX_TRANSMIT_WAIT seconds without any messages from the LwM2M Server, the LwM2M Client enters a sleep mode.

3. At some point in time the LwM2M Client wakes up again and transmits a registration update message. Note: During the time the LwM2M Client has been sleeping the IP address assigned to it may have been released and / or existing NAT bindings may have been released. If this is the case, then the client needs to re-run the DTLS handshake with the LwM2M Server since an IP address and/or port number change will destroy the existing security context. For performance and efficiency reasons it is RECOMMENDED to utilize the DTLS session resumption.

4. When the LwM2M Server receives a message from the Client, it informs determines whether any messages need to be sent to the LwM2M Client, as instructed by the application server.

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

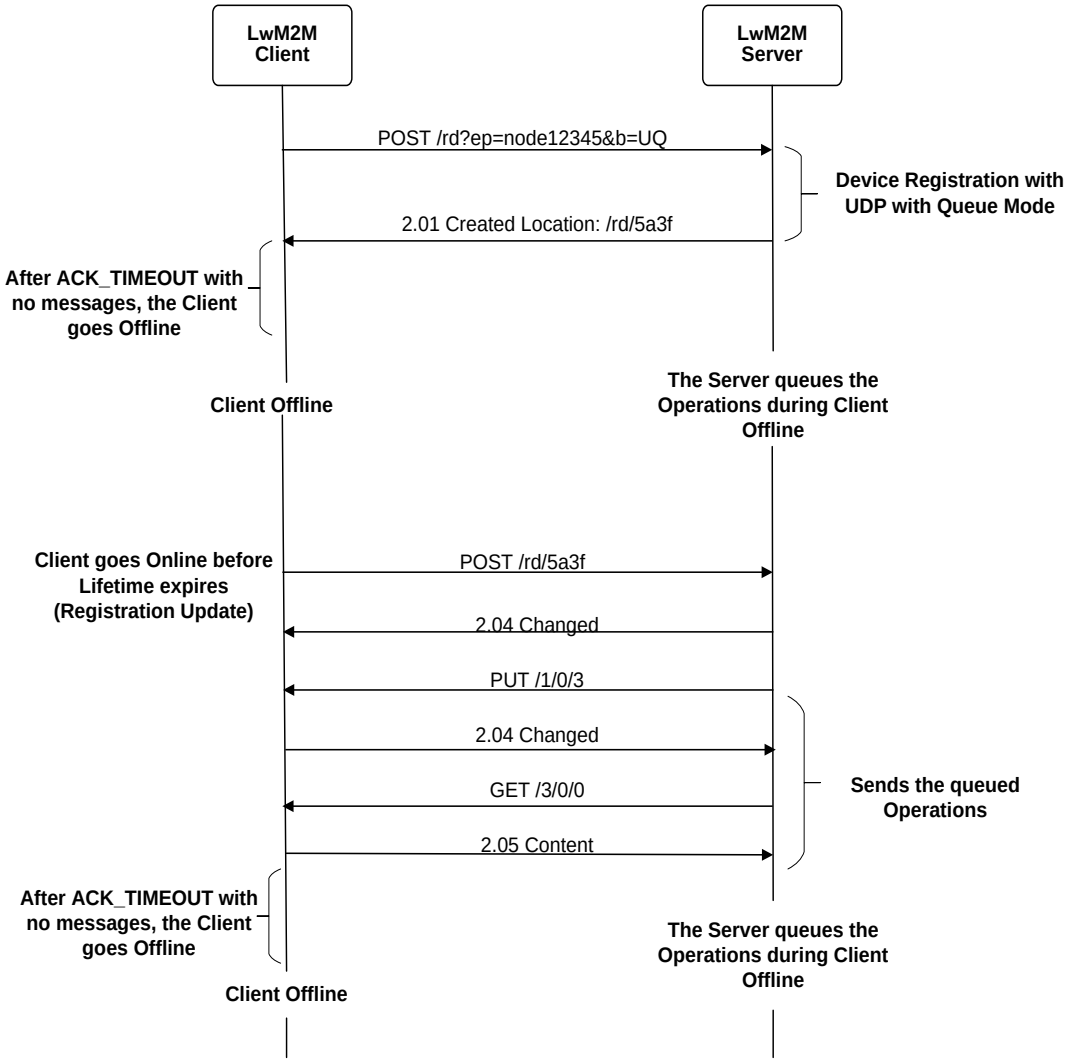


Figure 25: Example of Device Management & Service Enablement interface exchanges for Queue Mode

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

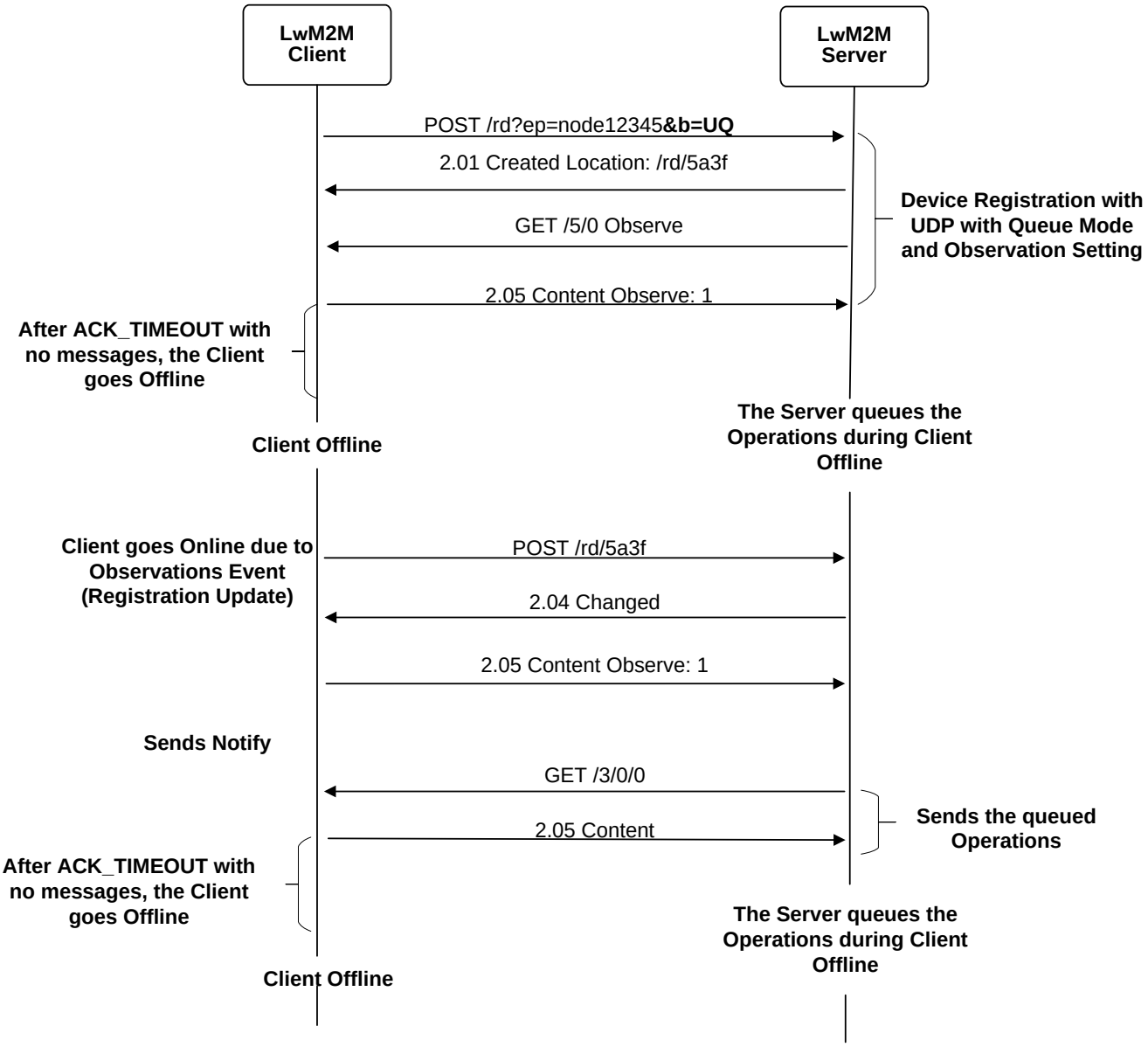
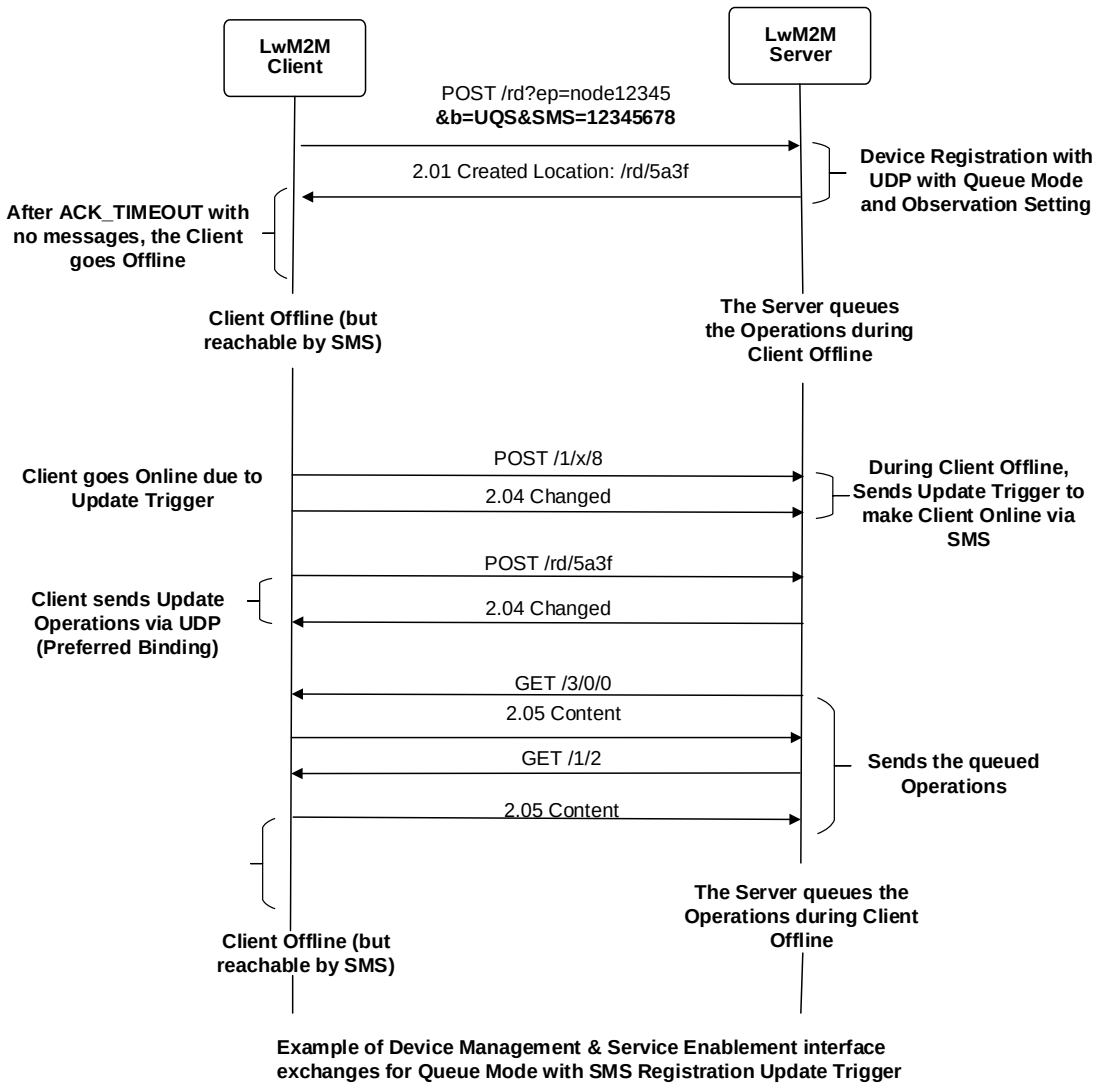


Figure 26: Example of an Information Reporting exchange for Queue Mode

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.



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

Figure 27: 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	Bootstrap-Request 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
Discover	2.05 Content	"Discover" operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.04 Not Found	URI of "Discover" operation is not found

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
	4.06 Not Acceptable	Inconsistent loaded configuration
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.
	4.12 Precondition Failed	Supported LwM2M Versions of the Server and the Client are not compatible
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
	2.31 Continue	
	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
	4.08 Request Entity Incomplete	
	4.13 Request entity too large	
	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 Cancel Observe	2.05 Content	Operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of Operation is not found or not supported
	4.05 Method Not Allowed	Target is not allowed for the Operation
	4.06 Not Acceptable	None of the preferred Content-Formats can be returned
Notify	2.05 Content	“Notify” operation is completed successfully

Table 27: Response Codes

If any operation in Table 21, Table 24 and Table 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
OMA-TS-LightweightM2M-V1_0-20170208-A	08 Feb 2017	Status changed to Approved by TP TP Ref # OMA-TP-2017-0009-INP_LightweightM2M-V1_0_ERP_for_Final_Approval

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 5.2.3.1	
LwM2M-BOOT-003-C-O	Support of Bootstrap from Smartcard	Section 5.2.3.2, Appendix F	LwM2M-BOOT-012C-O
LwM2M-BOOT-004-C-O	Support of Client Initiated Bootstrap	Section 5.2.3.3	
LwM2M-BOOT-005-C-O	Support of Server Initiated Bootstrap	Section 5.2.3.4	
LwM2M-BOOT-006-C-M	Support of LwM2M Server Bootstrap Information	Section 5.2.2	
LwM2M-BOOT-007-C-O	Support of LwM2M Bootstrap-Server Bootstrap Information	Section 5.2.2	
LwM2M-BOOT-008-C-M	Support of accepting Bootstrap Information transferred	Section 5.2.2	
LwM2M-BOOT-009-C-M	Support of Bootstrap Sequence	Section 5.2.4	
LwM2M-BOOT-010-C-M	Support of Bootstrap Security	Section 5.2.5	
LwM2M-BOOT-011-C-O	Support of Bootstrap from Smartcard with Secure Channel	Section 5.2.3.2, Appendix F	LwM2M-BOOT-012C-O AND LwM2M-SEC-007-C-O
LwM2M-BOOT-012-C-O	Retrieve & Process bootstrap data from Smartcard	Section 5.2.3.2	
LwM2M-BOOT-013-C-O	Check for Bootstrap Data change in Smartcard	Section 5.2.3.2	
LwM2M-BOOT-014-C-O	Support of the BOOSTRAP-REQUEST operation	Section 5.2.7.1	LwM2M-BOOT-004-C-O
LwM2M-BOOT-015-C-O	Support of the BOOTSTRAP-FINISH, BOOTSTRAP DISCOVER, BOOTSTRAP WRITE, BOOTSTRAP DELETE operations	Section 5.2.7.2 Section 5.2.7.3 Section 5.2.7.4 Section 5.2.7.5	LwM2M-BOOT-004-C-O OR LwM2M-BOOT-005-C-O
LwM2M-BOOT-016-C-O	Check and report Bootstrap Configuration Inconsistency	Section 5.2.6	LwM2M-BOOT-004-C-O OR LwM2M-BOOT-005-C-O

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-M	Support of LwM2M Version parameter	Section 5.3.1	
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 5.3.2	(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)
LwM2M-CR-0011-C-M	No Security Object (ID:0) and Security Object Instances in the parameters list	Section 5.3.1	
LwM2M-CR-0012-C-M	Support of Object Versioning in the Object and Object Instances parameter list	Section 5.3.1	

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 5.4.3	
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	

Item	Function	Reference	Requirement
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-O	Support of Plain Text format	Section 6.4, 6.4.1	
LwM2M-DF-002-C-O	Support of Opaque format	Section 0.6.4, 6.4.2	
LwM2M-DF-003-C-M	Support of TLV format	Section 0.6.4, 6.4.3	
LwM2M-DF-004-C-O	Support of JSON format	Section 0.6.4, 6.4.4	

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 7.1.7	
LwM2M-SEC-003-C-O	Support of Raw Public Key Certificates mode	Section 7.1.8	
LwM2M-SEC-004-C-O	Support of X.509 Certificates mode	Section 7.1.9	
LwM2M-SEC-005-C-O	Support of No Sec mode	Section 7.1.10	
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	

Item	Function	Reference	Requirement
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 5.2.3.4	
LwM2M-BOOT-010-S-M	Support of Bootstrap Security	Section 5.2.5	

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	

Item	Function	Reference	Requirement
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-008-S-M	Support of “Update” operation	Section 5.3.2	
LwM2M-CR-009-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 5.4.3	
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 6.4, 6.4.1	
LwM2M-DF-002-S-M	Support of Opaque format	Section 6.4, 6.4.2	
LwM2M-DF-003-S-M	Support of TLV format	Section 6.4, 6.4.3	
LwM2M-DF-004-S-M	Support of JSON format	Section 6.4, 6.4.4	

B.2.6 Security

Item	Function	Reference	Requirement
LwM2M-SEC-002-S-M	Support of Pre-Shared Keys mode	Section 7.1.7	
LwM2M-SEC-003-S-M	Support of Raw Public Key Certificates mode	Section 7.1.8	
LwM2M-SEC-004-S-M	Support of X.509 Certificates mode	Section 7.1.9	
LwM2M-SEC-005-S-M	Support of No Sec mode	Section 7.1.10	
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. For example, the integer value -750 results in the 4 characters/byte long ASCII string "-750".	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 numeric representation. For example, we use a floating point number with the significand of 6.667, a base of 10 and the exponent of -11. This represents the number 6.667e-11 in scientific notation and will be represented as "0.00000000006667" as an ASCII string.	Represented as a binary floating point value [IEEE 754-2008] [FLOAT]. 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 8 bit unsigned integer with the value 0 for False and the value 1 for True.	Represented as the ASCII value 0 or 1.	Represented as an 8 bit unsigned Integer with value 0, or 1. The Length of a Boolean value MUST always be 1 byte.
Opaque	A sequence of binary octets, the minimum and/or maximum length of the String MAY be defined.	Represented as a Base64 encoding of the binary data [RFC4648]. For example, the sequence of bytes (in hex notation) {0x01, 0x02, 0x03, 0x04, 0x05} converts to the ASCII string "AQIDBAU=" in Base64 encoding.	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. For example, 1476186613 seconds since Jan 01 1970, which represents Tuesday, 11-Oct-16 11:50:13 UTC, are represented as the ASCII string "1476186613", which has 10 characters/bytes.	Same representation as Integer.

Data Type	Description	Text Format	TLV Format
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 bit unsigned integers following the Network Byte Order convention. The Most Significant Half-word is an Object ID, the Least Significant Half-word is an Object Instance 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 two 16 bit unsigned integers one beside the other. The first one represents the Object ID, and the second one represents the Object Instance ID. This value is always 4 bytes long.
none	no specific data type affected to that resource: it exclusively concerns Executable Resource	Not applicable	Not applicable

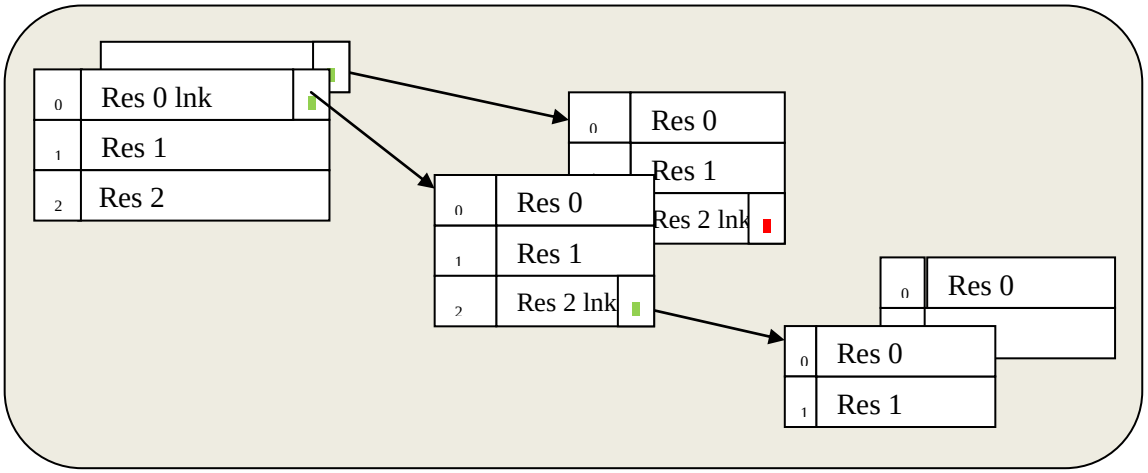


Figure 28: Object link Resource simple illustration

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}[:{version}]

- **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}[:{version}]" 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". The "version" field, follows the rules specified in Section 6.2 related to Object Versioning Policy.

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 any Instance of the Object that Resource belongs to, MUST instantiate such a Resource (refer Section 6.1, Resource Model). If this field is "Optional", then this Resource MAY be omitted from some - or even all - Instances of the Object that Resource belongs to.
- **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) encoded as an empty value in Object DDF file.
- **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	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, the Object ID and potentially the Object Version (see Section 6.2 Object Versioning) as follows:

urn:oma:lwm2m:{oma,ext,x}:{Object ID}[:{version}].

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 28: 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. The format of the CoAP URI is defined in Section 6 of RFC 7252.

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	0-4		Determines which UDP payload security mode is used 0: Pre-Shared Key mode 1: Raw Public Key mode 2: Certificate mode 3: NoSec mode 4: Certificate mode with EST
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 (Smartcard 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	1-65534		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. Specific ID:0 and ID:65535 values MUST NOT be used for identifying the LwM2M Server (Section 6.3).
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 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, or if this resource is not instantiated, 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. Since the security of the default PSK Cipher Suites rely on the length and the entropy of this shared secret it is RECOMMENDED to provision a 16 byte (128 bit) key or longer in the Secret Key Resource. Recommendations for generating random keys are provided in RFC 4086 [RFC4086].
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 public key MUST be encoded using the SubjectPublicKeyInfo structure, as described in [RFC7250].

The private key is encoded as defined in [RFC5958].

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, encoded using [RFC5958], as the private key in the RPK mode. The certificate in a DER encoded binary format, as defined in [RFC5280], is stored in 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 (see Section 5.3 Registration)
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" operations or temporarily 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 Notifications per 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 transport for the Current Binding 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 that transport for the Current Binding Mode.

E.3 LwM2M Object: Access Control

Description

Access Control Object is used to check whether the LwM2M Server has access right for performing an 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 20: LwM2M Identifiers.
2	ACL	RW	Multiple	Optional	Integer	16-bit		The Resource Instance ID MUST be the Short Server ID of a certain LwM2M Server for which associated access rights are contained in the Resource Instance value. The Resource Instance ID 0 is a specific ID, determining the ACL Instance which contains the default access rights. Each bit set in the Resource Instance value, grants an access right to the LwM2M Server to the corresponding 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 such an LwM2M Server can manage the Resources of this Object Instance. The specific value MAX_ID=65535 means this Access Control Object Instance is created and modified during a Bootstrap phase only.

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	Mandatory	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
1	Model Number	R	Single	Optional	String			A model identifier (manufacturer specified string)
2	Serial Number	R	Single	Optional	String			Serial Number
3	Firmware Version	R	Single	Optional	String			Current firmware version of the Device. The Firmware Management function could rely on this resource.
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 to go through initial deployment sequence where provisioning and bootstrap sequence is performed. This requires client ensuring post factory reset to have minimal information to allow it to carry out one of the bootstrap methods specified in section 5.2.3. 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 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 is 1).
10	Memory Free	R	Single	Optional	Integer		KB	Estimated current available amount of storage space which can store data and software in the LwM2M Device (expressed in kilobytes).
11	Error Code	R	Multiple	Mandatory	Integer	0-8		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.

<table>
<tr><td>16</td><td>Supported Binding and Modes</td><td>R</td><td>Single</td><td>Mandatory</td><td>String</td><td></td><td></td><td>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".</td></tr>
<tr><td>17</td><td>Device Type</td><td>R</td><td>Single</td><td>Optional</td><td>String</td><td></td><td></td><td>Type of the device (manufacturer specified string: e.g., smart meters / dev Class…)</td></tr>
<tr><td>18</td><td>Hardware Version</td><td>R</td><td>Single</td><td>Optional</td><td>String</td><td></td><td></td><td>Current hardware version of the device</td></tr>
<tr><td>19</td><td>Software Version</td><td>R</td><td>Single</td><td>Optional</td><td>String</td><td></td><td></td><td>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 of Software are together managed by LwM2M Firmware Update Object (Object ID 5)</td></tr>
<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>21</td><td>Memory Total</td><td>R</td><td>Single</td><td>Optional</td><td>Integer</td><td></td><td></td><td>Total amount of storage space which can store data and software in the LwM2M Device (expressed in kilobytes).</td></tr>
</table>

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.

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 Strength, 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 (as indicated by Resource 0 of this Object). For more details on Network Measurement Report, refer to the appropriate Cellular or Wireless Network standards. (e.g., for LTE Cellular Network refer to ETSI TS 36.133 specification).
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 Addresses	R	Multiple	Optional	String			The IP address of the next-hop IP router, on each of the interfaces specified in resource 4 (IP Addresses) 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_TS_23.003] and in [3GPP_TS_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_TS_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_TS_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_TS_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. The firmware update MAY require to reboot the device; it will depend on a number of factors, such as the operating system architecture and the extent of the updated software.

The envisioned functionality with LwM2M version 1.0 is to allow a LwM2M Client to connect to any LwM2M version 1.0 compliant Server to obtain a firmware imagine using the object and resource structure defined in this section experiencing communication security protection using DTLS. There are, however, other design decisions that need to be taken into account to allow a manufacturer of a device to securely install firmware on a device. Examples for such design decisions are how to manage the firmware update repository at the server side (which may include user interface considerations), the techniques to provide additional application layer security protection of the firmware image, how many versions of firmware imagines to store on the device, and how to execute the firmware update process considering the hardware specific details of a given IoT hardware product. These aspects are considered to be outside the scope of the LwM2M version 1.0 specification.

A LwM2M Server may also instruct a LwM2M Client to fetch a firmware image from a dedicated server (instead of pushing firmware imagines to the LwM2M Client). The Package URI resource is contained in the Firmware object and can be used for this purpose.
A LwM2M Client MUST support block-wise transfer [CoAP_Blockwise] if it implements the Firmware Update object.
A LwM2M Server MUST support block-wise transfer. Other protocols, such as HTTP/HTTPs, MAY also be used for downloading firmware updates (via the Package URI resource). For constrained devices it is, however, RECOMMENDED to use CoAP for firmware downloads to avoid the need for additional protocol implementations.

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	RW	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. The URI format is defined in RFC 3986. For example, coaps://example.org/firmware is a syntactically valid URI. The URI scheme determines the protocol to be used. For CoAP this endpoint MAY be a LwM2M Server but does not necessarily need to be. A CoAP server implementing block-wise transfer is sufficient as a server hosting a firmware repository and the expectation is that this server merely serves as a separate file server making firmware images available to LwM2M Clients.
2	Update	E	Single	Mandatory	none			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. Writing an empty string to Package Resource or to Package URI Resource, resets the Firmware Update State Machine: the State Resource value is set to Idle and the Update Result Resource value is set to 0. 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. Firmware Update mechanisms are illustrated below in Figure 29 of the LwM2M version 1.0 specification.
5	Update Result	R	Single	Mandatory	Integer	0-9		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 flash memory for the new firmware package. 3. Out of RAM during downloading process. 4: Connection lost during downloading process. 5: Integrity check failure for new downloaded package. 6: Unsupported package type. 7: Invalid URI 8: Firmware update failed 9: Unsupported protocol. A LwM2M client indicates the failure to retrieve the firmware imagine using the URI provided in the Package URI resource by writing the value 9 to the /5/0/5 (Update Result resource) when the URI contained a URI scheme unsupported by the client. Consequently, the LwM2M Client is unable to retrieve the firmware image using the URI provided by the LwM2M Server in the Package URI when it refers to an unsupported protocol.
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
8	Firmware Update Protocol Support	R	Multiple	Optional	Integer			This resource indicates what protocols the LwM2M Client implements to retrieve firmware images. The LwM2M server uses this information to decide what URI to include in the Package URI. A LwM2M Server MUST NOT include a URI in the Package URI object that uses a protocol that is unsupported by the LwM2M client. For example, if a LwM2M client indicates that it supports CoAP and CoAPS then a LwM2M Server must not provide an HTTP URI in the Packet URI. The following values are defined by this version of the specification: 0 – CoAP (as defined in RFC 7252) with the additional support for block-wise transfer. CoAP is the default setting. 1 – CoAPS (as defined in RFC 7252) with the additional support for block-wise transfer 2 – HTTP 1.1 (as defined in RFC 7230) 3 – HTTPS 1.1 (as defined in RFC 7230) Additional values MAY be defined in the future. Any value not understood by the LwM2M Server MUST be ignored.
9	Firmware Update Delivery Method	R	Single	Mandatory	Integer			The LwM2M Client uses this resource to indicate its support for transferring firmware images to the client either via the Package Resource (=push) or via the Package URI Resource (=pull) mechanism. 0 – Pull only 1 – Push only 2 – Both. In this case the LwM2M Server MAY choose the preferred mechanism for conveying the firmware image to the LwM2M Client.

E.6.1 Firmware Update State Machine

Figure 29 shows a possible implementation of the firmware update mechanism, described as a UML 2.0 state diagram. The state diagram consists of states, drawn as rounded rectangles, and transitions, drawn as arrows connecting the states. The syntax of the transition is trigger [guard] / behaviour. A trigger is an event that may cause a transition, guard is a condition and behaviour is an activity that executes while the transition takes place. The states additionally contain a compartment that includes assertions and variable assignments. For example, the assertion in the IDLE state indicates the value of the "Update Result" resource (abbreviated as "Res") must be between 0 and 9. The State resource is set to zero (0) when the program is in this IDLE state.

Errors during the Firmware Update process MUST be reported only by the "Update Result" resource. For instance, when the LwM2M Server performs a Write operation on resource "Package URI", the LwM2M Client returns a Success or Failure for the Write operation. Then if the URI is invalid, it changes its "Update Result" resource value to 7.

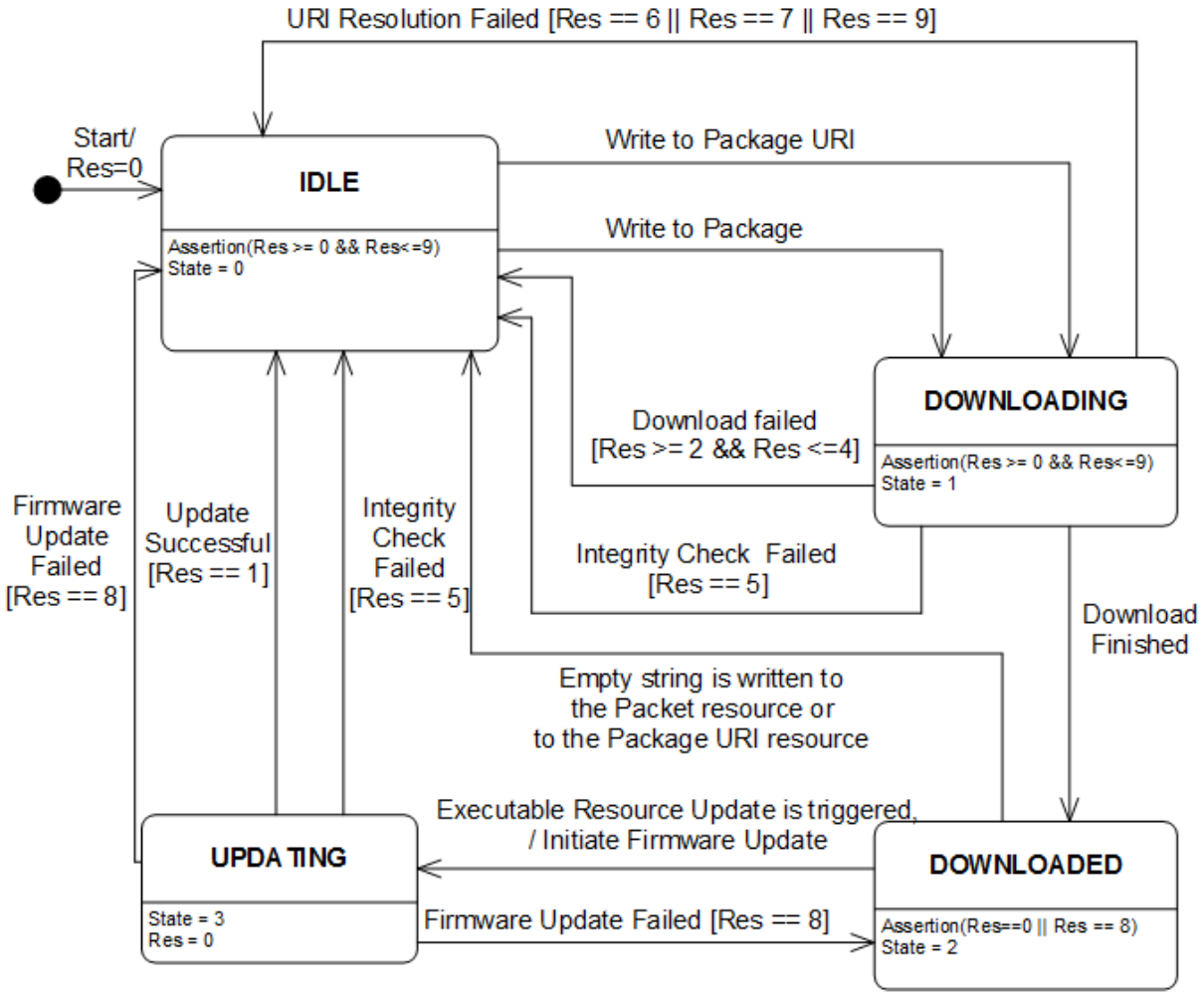


Figure 29: Firmware Update Mechanisms

E.6.2 Examples

The example depicted in Figure 30 illustrates a successful message exchange where a LwM2M Server pushes a firmware image to a LwM2M Client using the block-wise transfer. In this example the LwM2M Server indicates a block size of 128 bytes and the firmware image of 80 KiB (=81920 bytes) will be sent to the LwM2M Client in 640 messages with each 128 bytes payload. Since the LwM2M Server-provided block size matches the preferences of the LwM2M Client the exchange proceeds until the full firmware image is downloaded. In this example, no messages are lost during transmission.

```
LwM2M                                                    LwM2M
Server                                                   Client
  |                                                        |
  | CON [MID=1], POST, /5/0/0, 1:0/1/128       ------>     |
  |                                                        |
  | <------   ACK [MID=1], 2.31 Continue, 1:0/1/128        |
  |                                                        |
  | CON [MID=2], POST, /5/0/0, 1:1/1/128       ------>     |
  |                                                        |
  | <------   ACK [MID=2], 2.31 Continue, 1:1/1/128        |
  |                                                        |
  |               ... 637 exchanges ...                    |
```

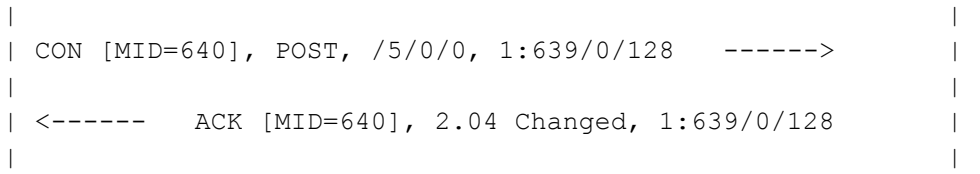


Figure 30: Example of a LwM2M Server pushing a firmware image to a LwM2M client

The second example shown in Figure 31 illustrates the case where the client was provided with a URI by the LwM2M Server (using the Package URI resource) and therefore fetches the firmware image from the indicated server. Note that only the retrieval of the firmware image from the LwM2M Server is shown in Figure 31 and not the initial configuration of the Package URI.

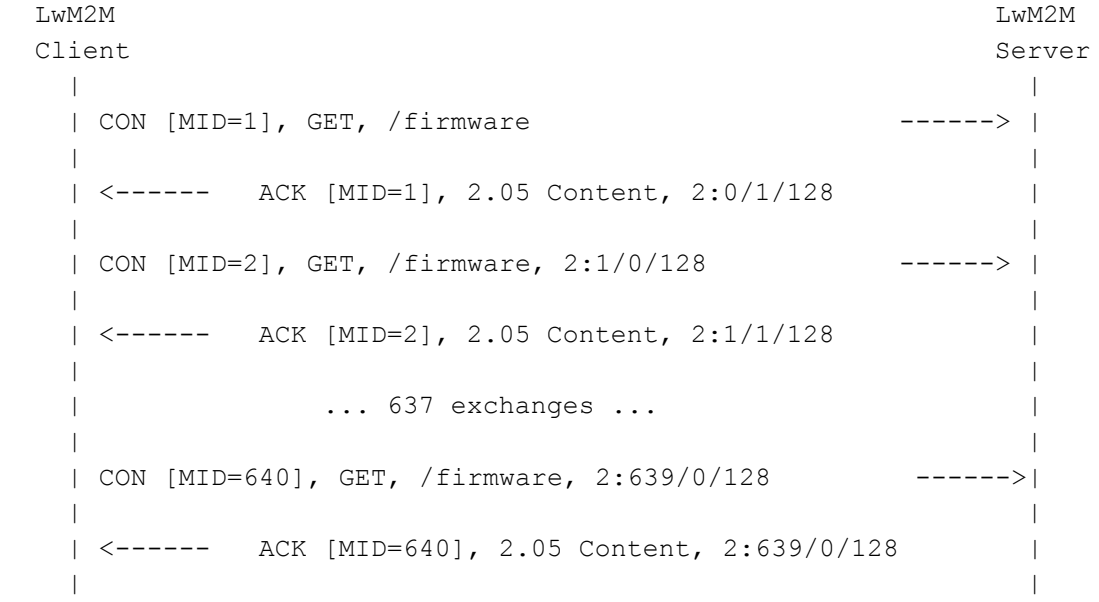


Figure 31: Example of a client fetching a firmware image

E.6.3 Firmware Update Consideration

If some Objects are not supported after firmware update, the LwM2M Client MUST delete all Object Instances of the Objects that are not supported.

E.7 LwM2M Object: Location

Description

The LwM2M Location Object provides the ability to express a point (in 2d, and in 3d), a circle (in 2d) and a sphere (in 3d). This location information is used to indicate where the LwM2M Client is located at the indicated point in time. A point is used when there is no known uncertainty and it forms part of a number of other geometries. A point may be specified using either world geodetic system (WGS) 84 (latitude, longitude) or WGS 84 (latitude, longitude, altitude). The circular area is used for coordinates in two-dimensional coordinate reference systems (CRSs) to describe uncertainty about a point. It is specified using a two-dimensional CRS (using latitude, longitude). The sphere is a volume that provides the same information as a circle in three dimensions. It is specified using a three-dimensional CRS (using latitude, longitude, and altitude). The use of the world geodetic system 1984 (WGS84) coordinate reference system and the usage of European petroleum survey group (EPSG) code 4326 for two-dimensional (2d) shape representations and EPSG 4979 for three-dimensional (3d) volume representations is mandated. For the circle and the sphere a confidence value of 95% is assumed [RFC7459]. Finally, the ability to convey information about moving objects is enabled by expression of speed and velocity.

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	Float		Deg	The decimal notation of latitude, e.g., -43.5723 [World Geodetic System 1984].
1	Longitude	R	Single	Mandatory	Float		Deg	The decimal notation of longitude, e.g., 153.21760 [World Geodetic System 1984].
2	Altitude	R	Single	Optional	Float		m	The decimal notation of altitude in meters above sea level.
3	Radius	R	Single	Optional	Float		m	The value in the Radius Resource indicates the size in meters of a circular area around a point geometry.
4	Velocity	R	Single	Optional	Opaque			The velocity of the LwM2M Client is defined in [3GPP-TS_23.032].
5	Timestamp	R	Single	Mandatory	Time			The timestamp of when the location measurement was performed.
6	Speed	R	Single	Optional	Float		Meters per second	Speed is the time rate of change in position of a LwM2M Client without regard for direction: the scalar component of velocity.

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 resources 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 do not reset resources 0-5.
8	Collection Period	RW	Single	Optional	Integer		Seconds	The default collection period in seconds. The value 0 indicates that the collection period is not set.

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), five accompanying Access Control Object Instances for those servers, a Device Object, a Connectivity Monitoring Object for a Cellular interface and a Firmware Update Object with no instance. The first Server controls the access control rights for both servers.

The Short Server ID 101 & 102, are respectively associated to the Server 1 and Server 2;

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	0
LwM2M Server Object [2]	1	1
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	0
Connectivity Monitoring Object	4	0
Firmware Update Object	5	-

Table 29: Object Instances of the example

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LwM2M Server URI	0		coaps://bootstrap.example.com	Example LwM2M Bootstrap-Server
Bootstrap-Server	1		true	
Security Mode	2		0	PSK mode
Public Key or Identity	3		b313cc22-f969-42ec-ad42	Example of a PSK Identity string
Server Public Key	4			Unused in PSK mode
Secret Key	5		e129791359950cbb1c8c3c582913b551	128-bit random sequence in hex encoding for use as a PSK.
Client Hold Off Time	11		3600	

Table 30: LwM2M Security Object [0]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LwM2M Server URI	0		coaps://server1.example.com	Example LwM2M Server 1
Bootstrap-Server	1		false	
Security Mode	2		0	PSK mode
Public Key or Identity	3		b313cc22-f969-42ec-ad42	Example of a PSK Identity string

Server Public Key	4			Unused in PSK mode
Secret Key	5		1ca2028d7197c778e9aef5ef69082bea	128-bit random sequence in hex encoding for use as a PSK.
Short Server ID	10		101	

Table 31: LwM2M Security Object [1]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
LwM2M Server URI	0		coaps://server2.example.com	Example LwM2M Server 2
Bootstrap-Server	1		false	
Security Mode	2		1	RPK mode
Public Key or Identity	3		3059301306072a8648ce3d020106082a8648ce3d03010703420004a09dcb1bc739e7f19afa9adcb1944968bfba73efcec29b50486eb44d1b29c193449970a3736830cb2bd5d7ec05d2bd1fc07b6df5e48b54ce77a6a7229c3a91c2	The raw public key of the LwM2M Client in ASN.1 DER format. The textual representation of the key can be found below labelled as *.
Server Public Key	4		3059301306072a8648ce3d020106082a8648ce3d0301070342000482c773f378d30c2783a48eb811b96cf6a90694a8564f1e7f6d366152f11698950f6f7c40cda585334f837750a102f5bf25508ceb9e55dfc0f12a455199a4c960	The raw public key of the LwM2M Server in ASN.1 DER format. The textual representation of the key can be found below labelled as **.
Secret Key	5		307702010104209f352da16495748e146fcb5370b8e96d292ced5567a8fae55a22bb67d91651b8a00a06082a8648ce3d030107a14403420004a09dcb1bc739e7f19afa9adcb1944968bfba73efcec29b50486eb44d1b29c193449970a3736830cb2bd5d7ec05d2bd1fc07b6df5e48b54ce77a6a7229c3a91c2	The private key of the client in PKCS#8 format. The textual representation of the key can be found below labelled as ***.
Short Server ID	10		102	

Table 32: LwM2M Security Object [2]

*: The client public key in text representation:

```
 0  89: SEQUENCE {
 2  19:   SEQUENCE {
 4   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
13   8:     OBJECT IDENTIFIER prime256v1 (1 2 840 10045 3 1 7)
      :     }
23  66:   BIT STRING
      :     04 A0 9D CB 1B C7 39 E7 F1 9A FA 9A DC B1 94 49
      :     68 BF BA 73 EF CE C2 9B 50 48 6E B4 4D 1B 29 C1
      :     93 44 99 70 A3 73 68 30 CB 2B D5 D7 EC 05 D2 BD
      :     1F C0 7B 6D F5 E4 8B 54 CE 77 A6 A7 22 9C 3A 91
      :     C2
      :   }
```

**: The server public key in text representation:

```
 0  89: SEQUENCE {
 2  19:   SEQUENCE {
 4   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
13   8:     OBJECT IDENTIFIER prime256v1 (1 2 840 10045 3 1 7)
      :     }
23  66:   BIT STRING
      :     04 82 C7 73 F3 78 D3 0C 27 83 A4 8E B8 11 B9 6C
      :     F6 A9 06 94 A8 56 4F 1E 7F 6D 36 61 52 F1 16 98
      :     95 0F 6F 7C 40 CD A5 85 33 4F 83 77 50 A1 02 F5
      :     BF 25 50 8C EB 9E 55 DF C0 F1 2A 45 51 99 A4 C9
      :     60
      :   }
```

***: The client private key in text representation:

```
 0 119: SEQUENCE {
 2   1:   INTEGER 1
 5  32:   OCTET STRING
      :     9F 35 2D A1 64 95 74 8E 14 6F CB 53 70 B8 E9 6D
      :     29 2C ED 55 67 A8 FA E5 5A 22 BB 67 D9 16 51 B8
39  10:   [0] {
41   8:     OBJECT IDENTIFIER prime256v1 (1 2 840 10045 3 1 7)
      :     }
51  68:   [1] {
53  66:     BIT STRING
      :       04 A0 9D CB 1B C7 39 E7 F1 9A FA 9A DC B1 94 49
      :       68 BF BA 73 EF CE C2 9B 50 48 6E B4 4D 1B 29 C1
      :       93 44 99 70 A3 73 68 30 CB 2B D5 D7 EC 05 D2 BD
      :       1F C0 7B 6D F5 E4 8B 54 CE 77 A6 A7 22 9C 3A 91
      :       C2
      :     }
      :   }
```

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 33: LwM2M Server Object [0]

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 34: LwM2M Server Object [1]

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		1	LwM2M Server Object 0 (Server 1)
Object Instance ID	1		0	
ACL	2	101	0b0000000000001111	For this Object Instance (1:0), Server 1 has 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 35: Access Control Object [0] (for the LwM2M Server Object Instance 0)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		1	LwM2M Server Object 1 (Server 2)
Object Instance ID	1		1	

ACL	2	102	0b0000000000001111	For this Object Instance (1:1), Server 2 has 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 36: Access Control Object [1] (for the LwM2M Server Object Instance 1)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		3	Device Object
Object Instance ID	1		0	
ACL	2	101	0b0000000000001111	For this Object Instance (3:0), Server 1 has access rights (R, W, E, D). Note that the Resource Instance ID indicates the Short Server ID.
ACL	2	102	0b0000000000000001	For this Object Instance (3:0), 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 37: Access Control Object [2] (for the Device Object Instance)

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	For this Object Instance (4:0), Server 1 has read-only access rights. Note that the Resource Instance ID indicates the Short Server ID.
ACL	2	0	0b0000000000000001	For this Object Instance (4:0), The other Servers except Server 1 have read-only access rights. Note that this Resource Instance ID indicates the default access rights.
Access Control Owner	3		101	Server 1 controls this Object Instance's access rights.

Table 38: Access Control Object [3] (for the Connectivity Monitoring Object Instance)

Resource Name	Resource ID	Resource Instance ID	Value	Notes
Object ID	0		5	Firmware Update Object
Object Instance ID	1		65535	Irrelevant

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 39: 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	16		U	UDP binding

Table 40: Device Object Instance

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	
Router IP Addresses	5	0	192.168.1.1	
Link Utilization	6		5	%
APN	7	0	internet	

Table 41: Connectivity Monitoring Object Instance

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

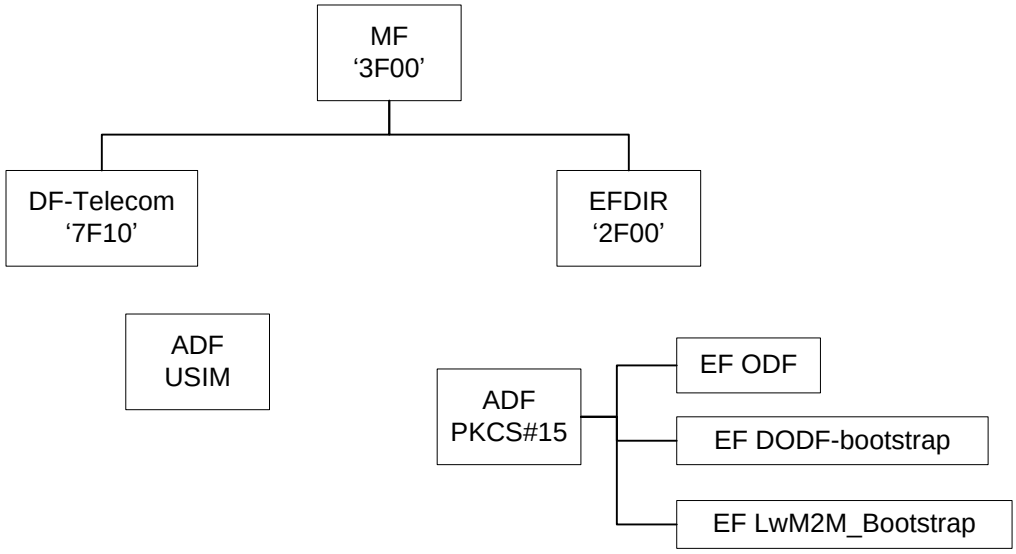


Figure 32: 3G UICC File Structure and Bootstrap data location

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] (i.e. 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 G.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 G.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 (§6.4.3), this section will only detailed how to store a collection of LwM2M Objects in this EF LwM2M_Bootstrap file; each Object is coded with a header containing a LwM2M Object ID and its Object Version coded in one or 2 Bytes, a LwM2M-TLV coding the Object Instances as payload, and a length being the size in bytes of this 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 full 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> <object_version> <length_of_object>
<lwm2m_object_instances>

<lwm2m_object_ID>             ::=  HWORD

<object_version>              ::=  IMPLICIT_VERSION | <other_version>

<other_version>               ::=  MAJOR_VERSION MINOR_VERSION  ; value %x0205 means version
2.5

<length_of_object>            ::=  HWORD

<lwm2m_object_instances>      ::= TLV data format as described in §0.6.46.4.3

HWORD                         ::= %x00-FFFF

IMPLICIT_VERSION               ::= %x00     ; means version 1.0 or the Object is defined in
the LwM2M Enabler

MAJOR_VERSION                 ::= %x01-FF

MINOR VERSION                 ::= %x00-FF
```

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 33):

- 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.

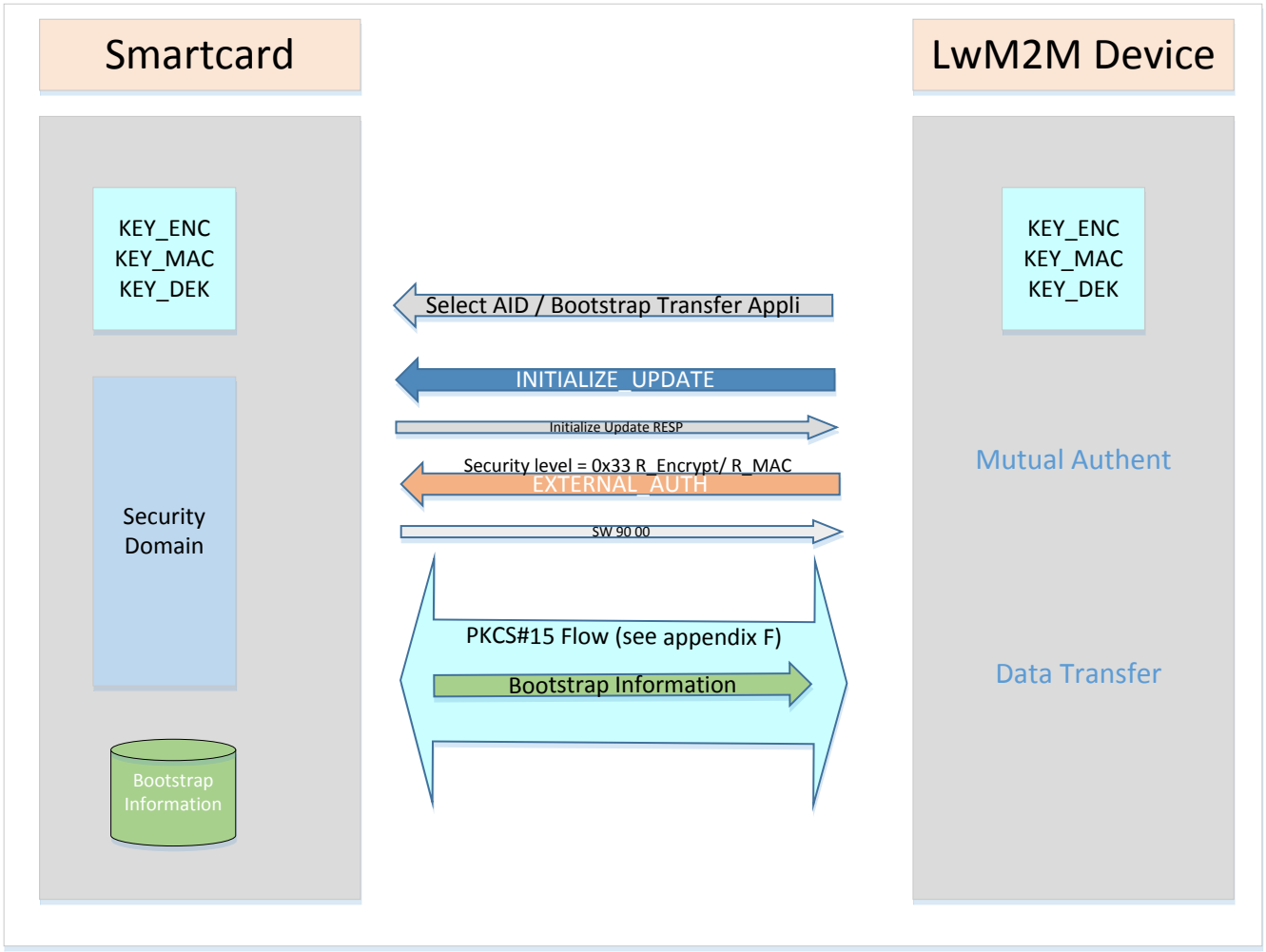


Figure 33: Bootstrap Information transfer from Smartcard to LwM2M Device using Secure channel according to [GLOBALPLATFORM] [GP SCP03] [GP AMD_A]

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. Media types

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

This section contains the IANA registration for media-type application/vnd.oma.lwm2m+tlv (see http://www.iana.org/assignments/media-types/media-types.xhtml).

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. The OMA LwM2M protocol uses source authentication and integrity protection.

This media type inherits the security issues associated with the TLV format.

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 6.4.3. 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): N/A

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

This section contains the IANA registration for media-type application/vnd.oma.lwm2m+json (see http://www.iana.org/assignments/media-types/media-types.xhtml).

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 6.4.4. 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): N/A

Appendix J. LwM2M Schema and Object Definition File (Informative)

For supporting the LwM2M Enabler version and the Object Version in the Definition file (.xml) of a LwM2M Object, the LwM2M schema (URL: ="http://openmobilealliance.org/tech/profiles/LWM2M.xsd") contains 2 optional elements: LwM2MVersion and ObjectVersion (see Appendix K).

The Object definition file (.xml), which describes the resources of a particular Object may contain these two elements:

a) the "LwM2MVersion" element indicates the minimum version of the LwM2M Enabler supporting that Object,

b) the "ObjectVersion" element indicates the version of the Object as defined in Section 6.2 "Object Versioning" in this document.

In addition:

- when the minimum LwM2M version supporting the Object is the Initial Version of the LwM2M Enabler (1.0), this information may be omitted.
- when the Object version is the Initial Version of that Object (1.0), the Object Version information may be omitted.
- when the Object definition is part of any LwM2M Enabler, the Object Version information may be omitted.

For example: an Object ID:44 in Version 2.4 for which the minimum LwM2M version supporting this Object Version is 1.1, will have an URN defined as "*urn:oma:lwm2m:oma:44:2.4"* used to name the associate Object definition (.xml) file on the OMNA portal:

```
<?xml version="1.0" encoding="UTF-8"?>
<LWM2M xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://openmobilealliance.org/tech/profiles/LWM2M.xsd" >
        <Object ObjectType="MODefinition">
                <Name>MyDevice</Name>
                <Description1>< ![CDATA[This LWM2M Object is my device]]></Description1>
                <ObjectID>44</ObjectID>
                <LWM2MVersion>1.1</LWM2MVersion>
                <ObjectVersion>2.4</ObjectVersion>
                <ObjectURN>urn:oma:lwm2m:oma:44:2.4</ObjectURN>
                <MultipleInstances>Single</MultipleInstances>
                <Mandatory>Mandatory</Mandatory>
                <Resources>
                    .
                    .
               </Resources>
               </Description2>
               </Resources>
        </Object>
</LWM2M>
```

Appendix K. LwM2M Schema

This appendix provides the content of the LightweightM2M schema.

The schema was created to support the LwM2M Editor Tool.
The following elements and attributes are used by the LwM2M Editor Tool but are not part of the LwM2M protocol:

- Description1, it is used to insert the definition of the Object
- Description2, it is used to insert any extra information at the end of the table that contains the Resources
- ObjectType, used to identify if the file contains an Object and Resources or only Resources

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="LWM2M">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="Object">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Name" type="xs:string" />
       <xs:element name="Description1" type="xs:string" />
       <xs:element name="ObjectID" type="xs:unsignedShort" />
       <xs:element name="ObjectURN" type="xs:string" />
       <xs:element name="LWM2MVersion" type="xs:string" minOccurs="0"/>
       <xs:element name="ObjectVersion" type="xs:string" minOccurs="0"/>
       <xs:element name="MultipleInstances" >
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:enumeration value="Multiple"/>
          <xs:enumeration value="Single"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="Mandatory" >
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:enumeration value="Mandatory"/>
          <xs:enumeration value="Optional"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="Resources">
        <xs:complexType>
         <xs:sequence>
          <xs:element maxOccurs="unbounded" name="Item">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="Name" type="xs:string" />
             <xs:element name="Operations" >
              <xs:simpleType>
               <xs:restriction base="xs:string">
                <xs:enumeration value="R"/>
                <xs:enumeration value="W"/>
                <xs:enumeration value="RW"/>
                <xs:enumeration value="E"/>
                <xs:enumeration value=""/>
               </xs:restriction>
```

```
            </xs:simpleType>
          </xs:element>
          <xs:element name="MultipleInstances" >
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="Multiple"/>
                <xs:enumeration value="Single"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="Mandatory" >
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="Mandatory"/>
                <xs:enumeration value="Optional"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="Type" >
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="String"/>
                <xs:enumeration value="Integer"/>
                <xs:enumeration value="Float"/>
                <xs:enumeration value="Boolean"/>
                <xs:enumeration value="Opaque"/>
                <xs:enumeration value="Time"/>
                <xs:enumeration value="Objlnk"/>
                <xs:enumeration value=""/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="RangeEnumeration" type="xs:string" />
          <xs:element name="Units" type="xs:string" />
          <xs:element name="Description" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="ID" type="xs:unsignedShort" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Description2" type="xs:string" />
</xs:sequence>
<xs:attribute name="ObjectType" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```