Lightweight Machine to Machine Technical Specification: Core

Approved Version: 1.1.1 - 2019 06 17

Open Mobile Alliance

OMA-TS-LightweightM2M_Core-V1_1_1-20190617-A

master: 25 Jun 2019 16:01:00 *rev: c41ccb4*

Table of Contents

Table of Figures

Table of Tables

1. Scope

This document, the LwM2M CORE technical specification, describes the LwM2M messaging layer. The LwM2M TRANSPORT specification [LwM2M-TRANSPORT], a companion specification, details the mapping of the messaging layer to selected transports. The separation between transport and messaging layer improves readability and simplify extending LwM2M to further transports in the future. The LwM2M messaging layer uses a RESTful design with several interfaces and a simple data model.

1.1. LwM2M version 1.1

This specification defines version 1.1 of the LwM2M protocol with the following new features:

- Enhancement of the LwM2M bootstrapping capabilities allowing for incremental upgrades.
- Improved support for Public Key Infrastructure (PKI) deployments.
- Introduction of enhanced registration sequence mechanisms by the LwM2M Client to LwM2M Server(s).
- Support for LwM2M over TCP/TLS to better support firewall and NAT traversal.
- Support for application layer security for LwM2M based on OSCORE
- Better support of LwM2M over Low Power WANs, including 3GPP CIoT & LoRaWAN.
- Extended LwM2M operations to enable Resource Instance level access.
- Performance improvement for retrieving and updating Resources of multiple objects.
- Support for JSON using SenML with CBOR serialization for compressed payload with highly efficient transmission.
- Addition of new data types.

For additional transport & security enhancements in LwM2M v1.1 please refer to [LwM2M-TRANSPORT].

1.2. LwM2M version 1.0

This document augments LwM2M version 1.0. Version 1.1 is backwards compatible to v1.0 with respect to mandatory features. LwM2M v1.0 offers 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

[LwM2M-TRANSPORT]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Transport Layer"
[LwM2M-TS_1.0]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification, Version 1.0.2", February 2018
[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_24.008]	3GPP TS 24.008 "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3"
[3GPP-TS_36.133]	3GPP TS 36.133 "Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management"
[3GPP-TS_36.214]	3GPP TS 36.214 "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements"
[3GPP-TS_44.018]	3GPP TS 44.018 "Mobile radio interface layer 3 specification; GSM/EDGE Radio Resource Control (RRC) protocol"
[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 - August 2016
[OSCORE]	G. Selander, J. Mattsson, F. Palombini, L. Seitz, "Object Security for Constrained RESTful Environments", draft-ietf-core-object-security-16, Mar. 2019.
[DynLink]	Z. Shelby, M. Koster, C. Groves, B. Zhu, B. Silverajan, "Dynamic Resource Linking for Constrained RESTful Environments", (work in progress) draft-ietf-core-dynlink-07, October 2018
[CoRE_Interface]	Z. Shelby, M. Vial, "CoRE Interfaces", (work in progress) draft-ietf-core-interfaces-11, March 2018
[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/

[PKCS#15]	"PKCS #15 v1.1: Cryptographic Token Information Syntax Standard", RSA Laboratories, June 6, 2000. URL:ftp://ftp.cert.dfn.de/pub/pca/docs/PKCS/ftp.rsa.com/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
[RFC5234]	"Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. January 2008, URL:http://www.ietf.org/rfc/rfc5234.txt
[RFC4122]	"A Universally Unique Identifier (UUID) URN Namespace", P. Leach, et al. July 2005, URL:http://www.ietf.org/rfc/rfc4122.txt
[RFC6690]	Shelby, Z. "Constrained RESTful Environments (CoRE) Link Format", RFC6690, Aug 2012.
[RFC7049]	C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.
[RFC7542]	"The Network Access Identifier", A. DeKok, May 2015,URL:https://tools.ietf.org/html/rfc7542.txt
[RFC7159]	T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014
[SENML]	C. Jennings, Z. Shelby, J. Arkko, A. Keranen, C. Bormann "Sensor Measurement Lists (SenML)", RFC 8428, July 2018
[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.
[SCRRULES]	Open Mobile Alliance, "SCR Rules and Procedures, Version 1.0", 19. September 2006, URL: http://member.openmobilealliance.org/ftp/Public_documents/iop/Permanent_documents/OMA-ORG-SCR_Rules_and_Procedures-V1_0-20060919-A.zip

Table: 2.1.-1 Normative References

2.2. Informative References

[3GPP TS 31.116]	3GPP TS 31.116 (V10.2.0): "Remote APDU Structure for (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"
[3GPP-TS_27.060]	3GPP TS 27.060 Packet domain; Mobile Station (MS) supporting Packet Switched services
[3GPP-TS_29.061]	3GPP TS 29.061 Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)
[3GPP-TR_23.720]	3GPP TR 23.720 Study on architecture enhancements for Cellular Internet of Things
[3GPP-TS_23.682]	3GPP TS 23.682 Architecture enhancements to facilitate communications with packet data networks and applications
[3GPP-TS_23.401]	3GPP TS 23.401 General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access

[3GPP-TS_24.301]	3GPP TS_24.301 Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3
[3GPP-TS_31.102]	3GPP TS 31.102 Characteristics of the Universal Subscriber Identity Module (USIM) application
[DMREPPRO]	"OMA Device Management Representation Protocol, Version 1.3". Open Mobile Alliance™. OMA-TS-DM_RepPro-V1_3. URL:http://www.openmobilealliance.org
[ETSI TS 102 226]	ETSI TS 102 226 (V11.0.0): "Smart cards; Remote APDU structure for UICC based applications (Release 11)"
[OMADICT]	"Dictionary for OMA Specifications", Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/
[OMNA]	"OMNA Lightweight M2M (LwM2M) Object & Resource Registry", *URL:http://www.openmobilealliance.org/*
[RFC3986]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.
[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

Table: 2.2.-1 Informative References

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

Object	An Object is a collection of logically related resources.
Object Instance	An Object Instance is one occurrence of an Object.
Resource	A Resource is an atomic unit of information.
Resource Instance	A Resource Instance is one occurrence of a Resource.
LwM2M Client	Component running on a device implementing the LwM2M protocol for interacting with the LwM2M Server and the LwM2M Bootstrap-Server. LwM2M Clients are often implemented in IoT devices and gateways.
LwM2M Server	Component implementing the server-side functionality of the LwM2M protocol for interacting with a LwM2M Client. Typically, the LwM2M Server software is running on a non-IoT device, such as an on-premise server or in a cloud-based infrastructure.
LwM2M Bootstrap-Server	The LwM2M Bootstrap-Server is a server responsible for provisioning essential information, including credentials, into the LwM2M Client to enable the LwM2M Client to perform the "Register" operation with one or more LwM2M Servers. Very often, the LwM2M Bootstrap-Server is the first LwM2M entity a LwM2M Client interacts with. The Bootstrap Interface is the only interface used between the LwM2M Client and the LwM2M Bootstrap-Server.
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

Table: 3.2.-1 Definitions

Note that wherever there is a LwM2M Security Object Instance there is potentially an associated LwM2M OSCORE Object Instance. Kindly consult [OMADICT] for more definitions used in this document.

3.3. Abbreviations

kB	Kilobyte; one kilobyte is 1000 bytes

CoAP	Constrained Application Protocol
DTLS	Datagram Transport Layer Security
LoRaWAN	LOng RAnge Wide Area Network
NB-IoT	NarrowBand Internet of Things
OCSP	Online Certificate Status Protocol
SMS	Short Message Service
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
OSCORE	Object Security for Constrained RESTful Environments

Table: 3.3.-1 Abbreviations

4. Introduction

This enabler defines the application layer communication protocol between a LwM2M Server and a LwM2M Client as well as between the LwM2M Bootstrap-Server and the LwM2M Client. The LwM2M Device includes a LwM2M Client component. 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 lightweight and compact protocol mechanisms, as well as an efficient resource data model.

The LwM2M messaging layer is inspired by the RESTful design model based on [CoAP]. Interactions are between clients and servers using request/response model. Uniform Resource Identifiers (URIs) are used to identify and interact with object and resources while the protocol allows stateless and layered architecture.

Four interfaces are designed between the three entities, as shown in the architecture in Figure: 4.-1 The overall architecture of the LwM2M Enabler:

- Bootstrap
- Client Registration
- Device Management and Service Enablement
- Information Reporting

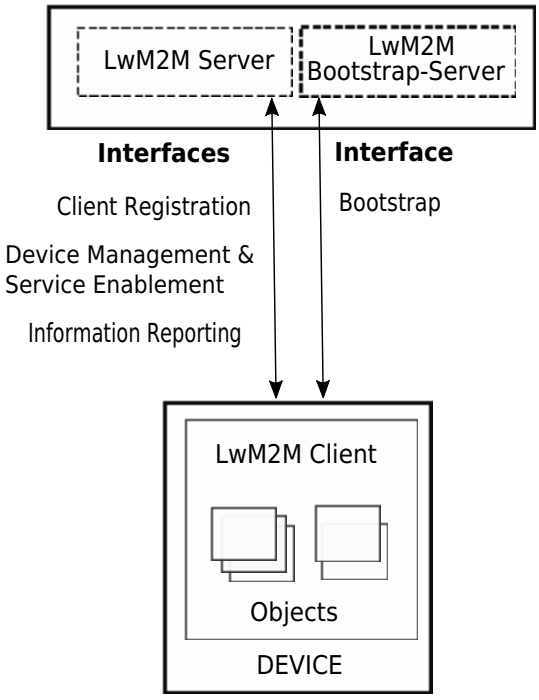


Figure: 4.-1 The overall architecture of the LwM2M Enabler

4.1. Version 1.1

Version 1.1 of LwM2M introduced the following objects, which are part of core specification:

21. OSCORE Object

Version 1.1 of LwM2M modified the following objects, which are part of core specification:

0. Security Object
1. Server Object

4.2. Version 1.0

Version 1.0 of LwM2M introduced the following objects, which are part of core specification:

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. Fundamental Considerations

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 carrying information only (e.g. Discover) to carrying parameters for setting up certain actions on the LwM2M Client (e.g. Notifications).

Attributes attached to Objects, Object Instances, Resources 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 assigned at various levels: Object, Object Instance, Resource levels. Additionally, precedence rules apply when the same Attribute receives a value at different levels.

The rules below govern the 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 4 different levels: the Resource Instance level, the Resource level, the Object Instance level and the Object level. Typically, an R-Attribute attached to a Multiple-Instance resource, can be set with an individual value to any Instance of such a Multiple-Instance Resource.

  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 3a: in the particular case this Resource is a Multiple-Instance Resource, the value of the R-Attribute set at the Resource Instance level will prevail over the value of the same Attribute set at the Resource 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 Table: 5.1.1.-1 Attribute Characteristics.

Attribute characteristic	Description
Name	Attribute Name used to reference a specific Attribute in that Enabler (e.g. "Minimum Period")
CoRE Link Param	the encoding used when this Attribute is transferred through a CoRE link parameter. See section 2 of [RFC6690]
Attachment	The Object, Object Instance, or Resource, to which an Attribute is logically applied
Assignation Level	The Level (Object, Object Instance, Resource, Resource Instance) where the value of the Attribute is set (by WRITE-ATTRIBUTES).
Class	Attributes are organized according to their purpose; 2 Class of Attributes are supported in LwM2M v1.1 <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. Data Types (Normative))
Value	The Value carried by this Attribute : its data type must be of "Value Type"

Table: 5.1.1.-1 Attribute Characteristics

Some Attributes MAY be exposed to the LwM2M Server in the payload response to a "Discover" operation (Section 6.3.2. Discover Operation).

The value of some Attributes MAY be changed by the LwM2M Server in using the "Write-Attributes" operation (Section 6.3.4. Write-Attributes Operation); which Attribute are concerned are marked as "W" (writable) in the table of the next section.

Note: A payload response to a "Discover" operation 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 the LwM2M Server for example easing data management.

The LwM2M Server and LwM2M Client SHOULD support Table: 5.1.2.-1 Class Attributes, except when specifically mentioned as not required.

Attribute Name	CoRE Link param	Attachment	Assignation Level	Support Required	Access Mode	Value Type	Default Value	Applicability	Notes
Dimension	"dim" "=" 1*DIGIT	Resource	Resource	YES (Client)	R	Integer [0:255]	-	Multiple-Instance Resource	Number of instances existing for a Multiple-Instance Resource
Short Server ID	"ssid" "=" 1*DIGIT	Object Instance	Object Instance	YES (Client)	R	Integer [1:65534]	-	Security Object	Provides the LwM2M Server Short ID information (see Section 6.1.7.3. Bootstrap-Discover Operation)
Server URI	"uri" "=" quoted-string	Object Instance	Object Instance	YES (Client)	R	String	-	Security Object	Provides the LwM2M Server URI information (see Section 6.1.7.3. Bootstrap-Discover Operation)
Object Version	"ver" "=" 1*DIGIT "." 1*DIGIT	Object	Object	YES (Server) YES (Client) when able to support Objects in version > 1.0	R	String	1.0 (Initial Version)		Provides the version of the associated Object. The rules governing the usage of this parameter are specified in 7.2.3. Object Definition and Object Version Usage and Table: 7.2.3.-1 Object Version usage rules.
Enabler Version	"lwm2m" "=" 1*DIGIT "." 1*DIGIT	None	None	YES (Server) YES (Client)	R	String	-		Provides the version of the supported LwM2M Enabler.

Table: 5.1.2.-1 <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.

Note: for readability reason in the remaining part of this section, the term "Resource" will be used to designate either a Single-Instance Resource or an Instance of a Multiple-Instance Resource, according to the nature of the considered Resource (i.e. Single- or Multiple-Instance Resource).

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.

Notification Conditions: the notification of a Resource value will be sent when:

- a valid Change Value Condition ("Greater Than", "Less Than", OR "Step") - if any is defined - AND the "Minimum Period" Timing Conditions are both fulfilled for that Resource OR
- the "Maximum Period" Timing Condition is fulfilled.

Additionally the following rules MUST be considered:

- a "Maximum Period" (pmax) applied to a Resource, that is smaller than "the Minimum Period" applied to the same Resource MUST be ignored for that Resource,
- Change Value Conditions are considered as "valid", if the 2 following rules related to the Attributes defined in the table below ("Greater Than", "Less Than", "Step") are respected:
  - ("lt" value < "gt" value)
  - ("lt" value + 2*"st" values <"gt" value)

When the "Change Value Conditions" Attributes are set in a single Write-Attributes operation, a coherency check of the 2 rules above MUST lead to reject that operation if these rules are violated.

The "Minimum Evaluation Period" (epmin) and "Maximum Evaluation Period" (epmax) values can be used to configure the device to perform reporting evaluations. After the expiry of epmin, the device MAY immediately perform an evaluation per the "Notification Conditions" above. After the expiry of epmax, the device MUST perform an evaluation per the "Notification Conditions". If both the epmin and epmax attributes are defined, the epmin must be less than the epmax.

The behavior of all Attributes SHOULD follow [DynLink] except epmin and epmax, which are only defined in this specification.

The LwM2M Server MUST support and LwM2M Client SHOULD support all the <NOTIFICATION> Class Attributes listed in Table: 5.1.2.-2 class Attributes.

Attribute Name	CoRE Link param	Attachment	Assignation Level	Required	Access Mode	Value Type	Default Value	Apply Condition
Minimum Period	"pmin" "=" 1*DIGIT	Resource	Resource Resource Instance Object Object Instance	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 notification of an observed Resource is supposed to be generated but it is before pmin expiry, notification MUST be sent as soon as pmin 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" "=" 1*DIGIT	Resource	Resource Resource Instance Object Object Instance	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 when set in the LwM2M Server Account or considered as 0 otherwise. The value of 0, means pmax MUST be ignored. The maximum period parameter MUST be greater than the minimum period parameter otherwise pmax will be ignored for the Resource to which such inconsistent timing conditions are applied.								
Greater Than	"gt" "=" 1*DIGIT ["." 1*DIGIT]	Resource	Resource Resource Instance	No	RW	Float	-	Numerical & Readable Resource

Notes: This "gt" Attribute defines a threshold high value. When this Attribute is present, the LwM2M Client MUST notify the Server each time the Observed Resource value crosses this threshold with respect to pmin parameter and valid "Change Value Conditions" (see Notification Conditions above).								
Less Than	"lt" "=" 1*DIGIT ["." 1*DIGIT]	Resource	Resource Resource Instance	No	RW	Float	-	Numerical & Readable Resource
Notes: This "lt" 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 this threshold with respect to pmin parameter and valid "Change Value Conditions" (see Notification Conditions above).								
Step	"st" "=" 1*DIGIT ["." 1*DIGIT]	Resource	Resource Resource Instance	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 pmin parameter and "Valid Value Conditions" (see notification Conditions above).								
Minimum Evaluation Period	"epmin" "=" 1*DIGIT	Resource	Resource Resource Instance Object Object Instance	No	RW	Integer	-	Readable Resource
Notes: The Minimum Evaluation Period Attribute indicates the minimum time in seconds the LwM2M Client MUST wait between two evaluations of reporting criteria. In the absence of this parameter, the Evaluation Minimum Period is not defined.								
Maximum Evaluation Period	"epmax" "=" 1*DIGIT	Resource	Resource Resource Instance Object Object Instance	No	RW	Integer	-	Readable Resource
Notes: The Maximum Evaluation Period Attribute indicates the maximum time in seconds the LwM2M Client MAY wait between two evaluations of reporting criteria . When the Maximum Evaluation Period expires after the previous evaluation, a new evaluation MUST occur. In the absence of this parameter, the Maximum Evaluation Period is not defined.								

Table: 5.1.2.-2 <NOTIFICATION> class Attributes

Examples illustrating Attributes setting are provided in Section Figure: 6.3.-1 Example flows of Device Management & Service Enablement Interface.

5.2. Compatibility

5.2.1. LwM2M Server and LwM2M Client Versions

The LwM2M Server MUST support a LwM2M Client of the same major and minor version. In the case of the current version, a LwM2M Server whose version is 1.1 MUST support a LwM2M Client whose version is 1.1.

The LwM2M Server SHOULD support all LwM2M Client versions of the same major version. Backward compatibility with a common major version is strongly recommended to support deployment scenarios where LwM2M Servers are updated before the LwM2M Clients. For example, a LwM2M Server whose version is 1.1 SHOULD support a LwM2M Client whose version is 1.0. Forward compatibility with a common major version is also recommended to support deployment scenarios where the LwM2M Clients are updated before the LwM2M Server. For example, a LwM2M Server whose version is 1.0 SHOULD support a LwM2M Client whose version is 1.1.

The LwM2M Server MAY support LwM2M Clients of an earlier major version. As an example, a LwM2M Server whose version is 2.0 MAY support a LwM2M Client whose version is 1.1. This type of backward compatibility is recommended, when it is possible, to support deployment scenarios where LwM2M Servers are updated before the LwM2M Clients.

5.2.2. Optional Objects and Object Versions

Operations initiated by the LwM2M Client to the LwM2M Server may address objects that are not supported by the LwM2M Server. Similarly, operations initiated by the LwM2M Server to the LwM2M Client may address objects that are not supported by the LwM2M Client. This may occur because the objects are not recognized or the object versions are not compatible.

To enable compatibility between LwM2M Clients and LwM2M Servers for optional objects or object versions that are not supported, the LwM2M Client or LwM2M Server MUST, when possible, silently ignore transactions targeted at those objects.

5.2.3. Optional Resources

Operations initiated by the LwM2M Client to the LwM2M Server may address optional resources that are not supported by the LwM2M Server. Operations initiated by the LwM2M Server to the LwM2M Client may address optional resources that are not supported by the LwM2M Client.

To enable compatibility between LwM2M Clients and LwM2M Servers for optional resources that are not supported, the LwM2M Client or LwM2M Server MUST, when possible, silently ignore transactions targeted at those resources.

6. Interfaces

According to Figure: 4.-1 The overall architecture of the LwM2M Enabler 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 into uplink operations and downlink operations. The operations for each interface are defined in this section, and then mapped to protocol mechanisms in the LwM2M Transport specification [LwM2M-TRANSPORT].

Figure: 6.-1 Bootstrap Interface shows the operation model for the "Bootstrap" interface. For this interface, the operations are an uplink operation named "Bootstrap-Request" and downlink operations named "Bootstrap-Discover", "Bootstrap-Write", "Bootstrap-Read","Bootstrap-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 Server(s). Once a "Bootstrap-Write" operation initiated on the "Bootstrap" interface by the LwM2M Server, 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 LwM2M Bootstrap-Server is addressing the Bootstrap Information to the LwM2M Client, the LwM2M Bootstrap-Server MUST inform the LwM2M Client when this transfer is over by sending a "Bootstrap-Finish" operation.

In addition to the bootstrapping interface the required information for the LwM2M Client to function with LwM2M Server(s) may also be configured during manufacturing (so-called Factory Bootstrap) or via a smartcard (so-called Smartcard Bootstrap).

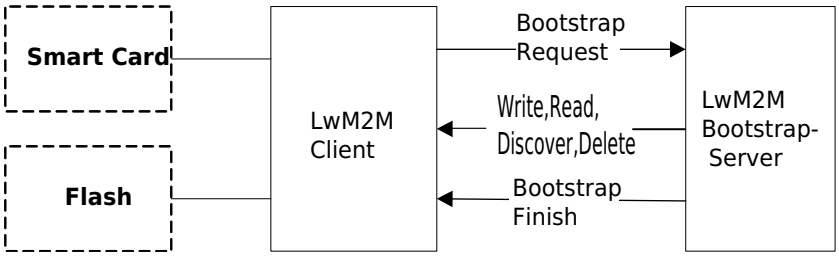


Figure: 6.-1 Bootstrap Interface

Figure: 6.-2 Client Registration shows the operation model for the interface "Client Registration". For this interface, the operations are uplink operations named "Registration", "Update" and "De-register".

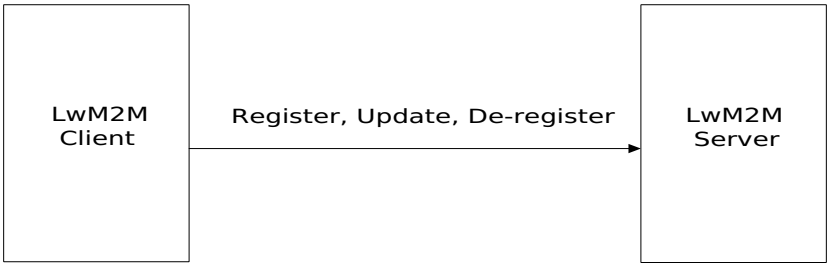


Figure: 6.-2 Client Registration

Figure: 6.-3 Device Management and Service Enablement shows the operational model for the "Device Management and

Service Enablement" interface. For this interface, the downlink operations are "Read", "Read-Composite", "Create", "Delete", "Write", "Write-Composite", "Execute", "Write-Attributes", and "Discover". These operations are used to interact with Resources, Resource Instances, Objects, Object Instances and/or their attributes exposed by the LwM2M Client. The "Read" and "Read-Composite" operations are 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" and "Write-Composite" operations are used to update values. The "Write-Attributes" operation is used to change attribute values. The "Execute" operation is used to initiate an action. The "Create" and "Delete" operations are used to create and delete Instances, respectively.

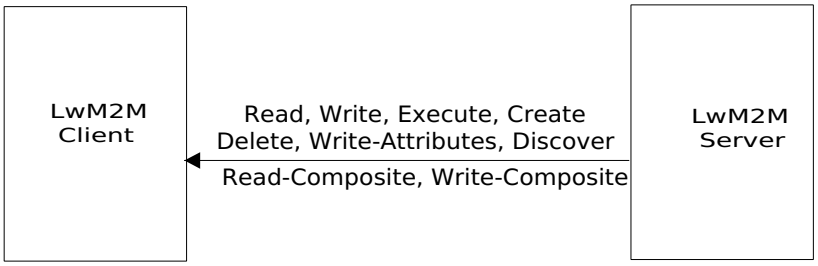


Figure: 6.-3 Device Management and Service Enablement

Figure: 6.-4 Information Reporting shows the operational model for the "Information Reporting" interface. For this interface, the downlink operations are "Observe", "Observe-Composite", "Cancel Observation", and "Cancel Observation-Composite". The "Notify" is an uplink operation, which is used to send a new value of a Resource from the LwM2M Client to the LwM2M Server. The "Send" operation is another uplink operation which is used by the LwM2M Client to send data to the LwM2M Server.

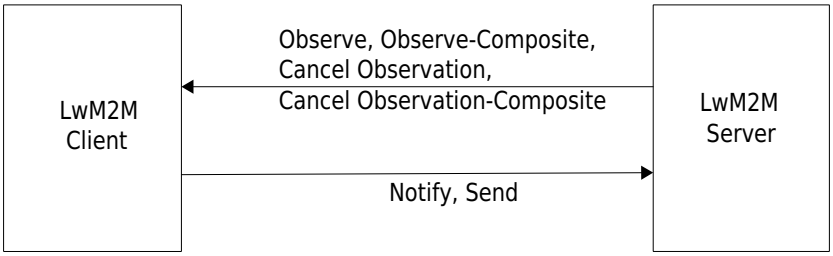


Figure: 6.-4 Information Reporting

The relationship between operations and interfaces is listed in Table: 6.-1 Relationship of operations and interfaces.

Interface	Direction	Operation
Bootstrap	Uplink	Bootstrap-Request
Bootstrap	Downlink	Bootstrap-Write, Bootstrap-Read, Bootstrap-Discover, Bootstrap-Delete, Bootstrap-Finish
Client Registration	Uplink	Register, Update, De-register
Device Management and Service Enablement	Downlink	Create, Read, Read-Composite, Write, Delete, Execute, Write-Attributes, Write-Composite, Discover

Information Reporting	Downlink	Observe, Observe-Composite, Cancel Observation, Cancel Observation-Composite
Information Reporting	Uplink	Notify,Send

Table: 6.-1 Relationship of operations and interfaces

6.1. Bootstrap Interface

The Bootstrap Interface is used to provision essential information into the LwM2M Client to enable the LwM2M Client to perform the "Register" operation with one or more LwM2M Servers.

There are four bootstrap modes supported by the LwM2M Enabler:

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

The 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). As specified in Section 6.1.3.4. Server Initiated Bootstrap, the "Server Initiated Bootstrap" mode is a method to invoke the "Client Initiated Bootstrap" mode.

The LwM2M Client MUST support at least one bootstrap mode specified in the Bootstrap Interface.

The LwM2M Bootstrap-Server MUST support the "Client Initiated Bootstrap" mode 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. The LwM2M TRANSPORT specification [LwM2M-TRANSPORT] provides further security-relevant information concerning the bootstrap techniques.

6.1.1. 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.

Note: During the Bootstrap Phase the LwM2M Client MAY ignore requests and flush all pending responses not related to the Bootstrap sequence.

6.1.2. Bootstrap Information

This section specifies the information that needs to be configured for a LwM2M Client to connect to LwM2M Server(s) or to the LwM2M Bootstrap-Server. This Bootstrap Information can be available before performing the Bootstrap Sequence

described in Section 6.1.4. Bootstrap Sequence 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

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 + potential OSCORE Object Instance)	No

Table: 6.1.2.-1 Bootstrap Information List

The LwM2M Client MUST have the LwM2M Server Bootstrap Information after the bootstrap sequence specified in Section 6.1.4. Bootstrap Sequence. The LwM2M Server Bootstrap Information is used by the LwM2M Client to register and connect to the LwM2M Server.

The LwM2M Client SHOULD have the LwM2M Bootstrap-Server Bootstrap Information. The LwM2M Server Bootstrap Information MUST contain at least one 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. Note also that a Security Object Instance may potentially be associated to an OSCORE Object Instance by means of an Object Link (Resource 17 of the Security Object Instance).

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.

(*) The LwM2M Client MUST have at least one LwM2M Server Account after completion of Bootstrap Sequence specified in Section 6.1.4. Bootstrap Sequence.

Please note that the LwM2M Client MUST accept Bootstrap Information sent via Bootstrap Interface without applying access control, as specified in Section 8.2. Authorization.

6.1.3. Bootstrap Modes

The following sub-sections provide further information for the four Bootstrap modes.

6.1.3.1. Factory Bootstrap

In this mode, the LwM2M Client has been configured with the necessary bootstrap information prior to deployment of the device. The configured information may be the LwM2M Bootstrap-Server Bootstrap Information and/or the LwM2M Server Bootstrap Information.

6.1.3.2. 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. Storage of LwM2M Bootstrap Information on the Smartcard (Normative). 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 sensitive 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 Section Appendix H. Secure channel between Smartcard and LwM2M Device Storage for secure Bootstrap Data provisioning (Normative).

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 the LwM2M Client, each time a "Register" or "Update" operation take place, with a LwM2M Server provisioned from Smartcard. As usual, the Bootstrap security rules (see Section 6.1.5. Bootstrap Security) then apply.

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

6.1.3.3. Client Initiated Bootstrap

The "Client Initiated Bootstrap" mode provides a mechanism for the LwM2M Client to retrieve Bootstrap Information from a LwM2M Bootstrap-Server. The "Client Initiated Bootstrap" mode requires a LwM2M Bootstrap-Server Account preloaded in the LwM2M Client.

At a minimum a LwM2M Client needs to have DTLS/TLS and/or OSCORE [OSCORE] security credentials preloaded to authenticate to a LwM2M Bootstrap-Server.

Figure: 6.1.3.3.-1 Client Initiated Bootstrap depicts protocol exchange graphically.

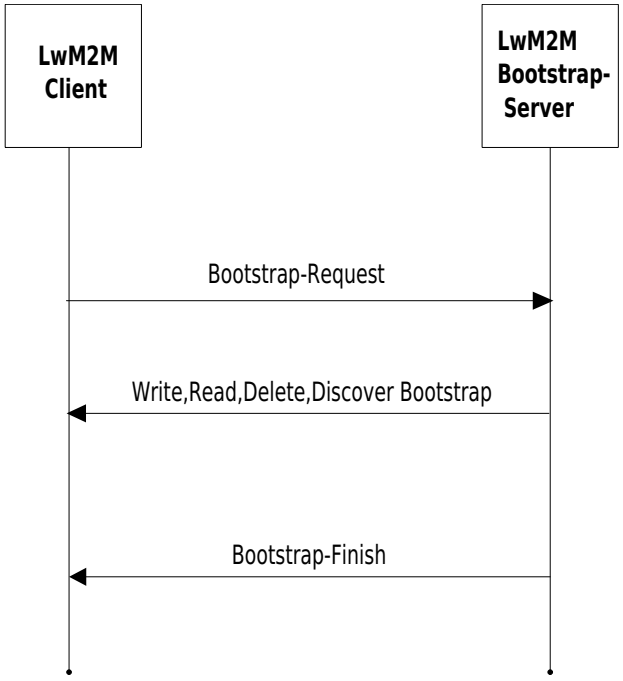


Figure: 6.1.3.3.-1 Client Initiated Bootstrap

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 SHOULD send 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. The LwM2M Client MAY omit "Endpoint Client Name" if it is equal to the identifier utilized in the security protocol.

Step #1: Configure Bootstrap Information

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

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 Bootstrap Information is OPTIONAL.

Step #2: Bootstrap-Finish

When the LwM2M Server has finished sending the Bootstrap Information to the LwM2M Client, the Server MUST send the "Bootstrap-Finish" operation to the Client to properly end this phase.

The bootstrap procedure failed when the LwM2M Client did not receive the "Bootstrap-Finish" operation after the EXCHANGE_LIFETIME time period expired. The EXCHANGE_LIFETIME parameter is defined in RFC 7252 [CoAP].

Step #3: Clean-up after successful Bootstrapping

Successful Bootstrapping means that the Bootstrap-Finish operation has been received by the LwM2M Client and the loaded configuration is considered consistent by the LwM2M Client. In this case the Bootstrap-Finish response code sent to the LwM2M Bootstrap-Server is 2.04 (Changed). If Bootstrapping was unsuccessful, the Bootstrap-Server Account MUST retain the values it had before the unsuccessful Bootstrapping sequence started and further statements

below in Step #3 do not 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 LwM2M Object: LwM2M Security).

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 "Not Acceptable" Response 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.

6.1.3.4. Server Initiated Bootstrap

In this mode the decision to trigger a Bootstrap Sequence is made by an authorized LwM2M Server. The LwM2M Server triggers the LwM2M Client to enter the "Client Initiated Bootstrap" mode rather than initiating a different protocol for configuring the Bootstrap Information in the LwM2M Client.

The trigger mechanism is performed through the execution of the "Bootstrap-Request Trigger" Resource available from the related Server Object Instance. A connection between a LwM2M Client and a LwM2M Server must already exist to enter this "Server Initiated Bootstrap" mode.

Note: proprietary mechanisms can be used to cause a LwM2M Client to enter the standard "Client Initiated Bootstrap" mode, but that is outside the scope of this specification.

The figure below depicts the "Server Initiated Bootstrap" flow initiated by an authorized LwM2M Server.

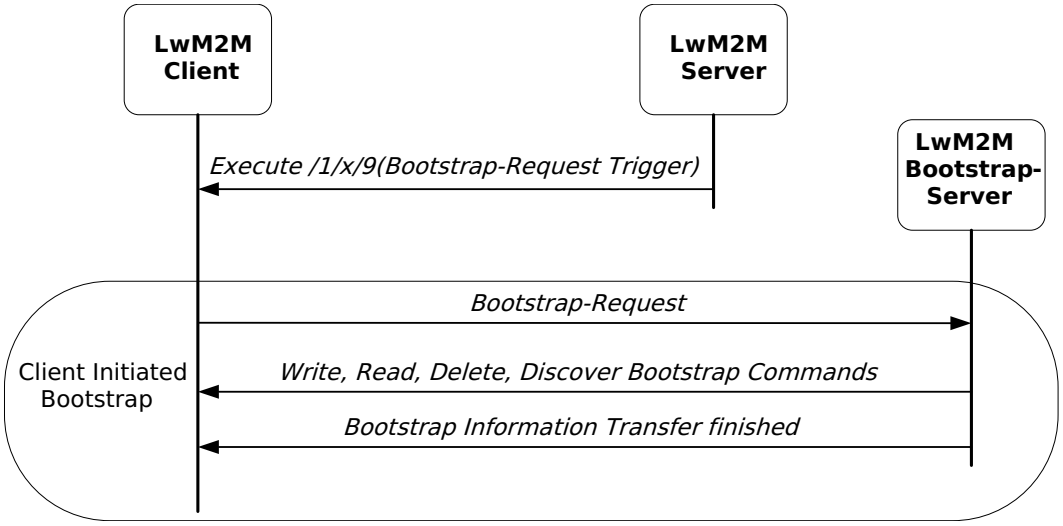


Figure: 6.1.3.4.-1 Procedure of Server Initiated Bootstrap initiated by an authorized LwM2M Server

6.1.4. 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 the LwM2M Client fails to register to all the LwM2M Servers or the Client doesn't have any LwM2M Server Object Instances, the LwM2M Client performs the Client Initiated Bootstrap.
5. A Server Initiated Bootstrap attempt (e.g. for updating a LwM2M Server Account) remains possible, but only if the LwM2M Client retains the corresponding LwM2M Bootstrap-Server Account.

6.1.5. Bootstrap Security

The information conveyed through the Bootstrap Interface is sensitive and requires communication security to be used. The security requirements for the Bootstrap Interface is discussed in [LwM2M-TRANSPORT].

6.1.6. 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 operation.

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 operation, the Bootstrap-Server MAY take corrective actions before issuing a new Bootstrap-Finish operation.

As specified in Section 6.1.3. Bootstrap Modes, the LwM2M Client MUST consider the Bootstrap procedure is failed when the LwM2M Client didn't receive a Bootstrap-Finish operation in a certain period (EXCHANGE_LIFETIME).

6.1.7. Bootstrap Operations

The mapping to underlying transports is detailed in [LwM2M-TRANSPORT].

The Bootstrap operations are used to help the Bootstrap-Server of setting a proper configuration in a LwM2M Client especially in the case of an incremental Bootstrap procedure for which a particular care must be observed to preserve the Client consistency (e.g. adding a new Server Account without breaking the access rights already in place in the targeted LwM2M Client).

6.1.7.1. Bootstrap-Request Operation

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 parameters:

Parameter	Required	Default Value	Notes
Endpoint Client Name	No	-	Indicates the LwM2M Client's "Endpoint Client Name". This parameter is optional if it is equal to the identifier in the security protocol and deployment is such that the security protocol identifier is always available for the server (e.g. no proxies in between).
Preferred Content Format	No	-	Indicates the numeric ID of the LwM2M Client's preferred Content Format for bootstrap configuration. The Content Format ID MUST be one of the SenML JSON, SenML CBOR, or TLV formats defined in Section 7.4. Data Formats for Transferring Resource Information.

Table: 6.1.7.1.-1 Bootstrap-Request Parameters

6.1.7.2. Bootstrap-Finish Operation

The Bootstrap-Finish operation is performed to terminate the Bootstrap Sequence previously initiated in "Client Initiated Bootstrap" mode (or in "Server Initiated Bootstrap" mode by extension).

This operation informs the LwM2M Client, that all the Bootstrap Information have been provided by the LwM2M Bootstrap-Server.

6.1.7.3. Bootstrap-Discover Operation

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

The "Bootstrap-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 list of Security Object Instances (ID:0) is reported, while it is not accessible in Device Management and Service Enablement Interface. If OSCORE is supported on the client, a list of OSCORE Objects (ID:21) is also reported, since this is not reported in Device Management and Service Enablement Interface either. 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 to the Object Version, see Section 7.2. Object Versioning and Table: 5.1.2.-1 Class Attributes - a sub-list of the Instances of such an Object Instance. In order to fully identify the Server Accounts supported by the Client, each element of the Instances list of the Security Object (Object ID:0) includes the associated Short Server ID and LwM2M Server URI in its parameters list while the elements of the Instances list of the Server Object (Object ID:1) also report the associated Short Server ID in their parameters list.

The Bootstrap-Server Security Object Instance does not include a Short Server ID in its parameter list (none is defined); therefore the LwM2M Server URI for the Bootstrap Server can also be omitted in such a parameters list.

In "Bootstrap-Discover" operation the targeted path '/' is accepted in place of the Object ID, for informing the Client to report all existing Object Instances.

The "Bootstrap-Discover" operation has the following parameters:

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

Table: 6.1.7.3.-1 Bootstrap-Discover Parameters

For example:

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

  </>;lwm2m=1.1,</0/0>;ssid=101;uri="coaps://server_1.example.com", </0/1/>, </0/2>;ssid=102;uri="coaps://server_2.example.com"

  with the meaning three LwM2M Servers are supported in that Client (which support the LwM2M Enabler in version 1.1), while the Instance ID:1 of the Security Object ID:0 contains the credentials for the LwM2M Bootstrap-Server.

- when the "Bootstrap-Discover" operation targets an Object with '/' the response to the operation could be:

  </>;lwm2m=1.1,</0/0>;ssid=101;uri="coaps://server_1.example.com", </0/1>, </1/0/>;ssid=101, </3/0>,</5>,</4>

  with the meaning the Client is supporting (in LwM2M version 1.1) 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 two other Objects, which are not instantiated yet (Firmware Update /5, and Connectivity Monitory Object /4).

- when the "Bootstrap-Discover" addresses a LwM2M Client supporting the LwM2M Enabler in release 1.1, and containing a configuration with Objects with Object ID of 0 (Security Object) and Object ID of 21 (OSCORE Object):

</>;lwm2m=1.1,</0/0>;ssid=101;uri="coaps://server_1.example.com",
</21/0>;ssid=101;uri="coap://server_1.example.com",</0/1>,</21/1/>

with the meaning 2 LwM2M Servers are supported in that Client (that supports the LwM2M Enabler in version 1.1), the Instance ID:0 of the Security Object ID:0 and the Instance ID:0 of the OSCORE Object ID:21 contains the DTLS and OSCORE credentials, respectively, of the the LwM2M server with ssid=101, while the Instance ID:1 of the OSCORE Object ID:21 contains the OSCORE credentials for the LwM2M Bootstrap-Server.

- when the "Bootstrap-Discover" addresses a LwM2M Client supporting the LwM2M Enabler in release 1.1, and containing a configuration with an hypothetical LwM2M Object ID:55 in Version 1.9, with one Instance (/55/0):

  </>;lwm2m=1.1,</0/0>,</0/1>;ssid=101;uri="coaps://server_1.example.com", </1/0/>;ssid=101, </3/0>,</5>,</4>,
  </55>;ver=1.9,</55/0>

  Note: In previous versions of this document, the LwM2M Enabler Version link parameter was not associated to any URI-Reference in the examples (e.g. "lwm2m=1.0,</0/0>;ssid=101,</0/1/>,</0/2>;ssid=102"). Thus, the provided examples were not in conformance with the ABNF of the CoRE Link Format RFC [RFC6690]. LwM2M Servers MAY accept this particular error in the LwM2M Client's response to the "Bootstrap-Discover" operation.

6.1.7.4. Bootstrap-Read Operation

The "Bootstrap-Read" operation in the Bootstrap Interface is a restricted form of the "Read" operation found in the Device Management and Service Enablement interface, and MUST be limited to target Objects that are strictly necessary to setup a proper configuration in a LwM2M Client.

In this specification the only acceptable targets for the "Bootstrap-Read" operation is the LwM2M Server Object (Object ID: 1) and the Access Control Object (Object ID:2). This operation will allow the Bootstrap-Server to query the existing Server Account(s) to determine validity and to add new Server Account(s) without breaking the access control rights already in place in the targeted LwM2M Client.

The "Bootstrap-Read" operation is used to access a single Instance or all Instances of the Server Object (ID:1) and the Access Control Object (ID:2).

The "Bootstrap-Read" operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object. In LwM2M 1.1, the Object ID MUST be '1' (Server Object) or '2' (Access Control Object). An error is reported otherwise.
Object Instance ID	No	-	Indicates the Object Instance to be read. If no Object Instance ID is indicated, then all Instances of the targeted Object are returned.

Table: 6.1.7.4.-1 Bootstrap-Read Parameters

As a simple illustration based on the second example of Section 7.4.4.2. Multiple Object Instance Request Examples: the "Bootstrap-Read /2" operation with JSON requested format could provide the following answer:

```
[{"bn":"/2/","n":"0/0","v":1},
{"n":"0/1","v":0},
{"n":"0/2/127","v":7},
{"n":"0/3","v":127},
{"n":"2/0","v":3},
{"n":"2/1","v":0},
{"n":"2/2/127","v":7},
{"n":"2/2/310","v":7},
{"n":"0/3","v":127}]
```

Table: 6.1.7.4.-2 Bootstrap-Read Example with Multiple Object Instances

The LwM2M Server MUST silently ignore optional resources included in the payload that are not supported.

6.1.7.5. Bootstrap-Write Operation

The "Bootstrap-Write" operation in Bootstrap Interface is different from the "Write" Operation in the 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 "Bootstrap-Write" operation targets an Object or an Object Instance, the LwM2M Client MUST silently 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 "Bootstrap-Write" operation can be sent multiple times.

Only in the Bootstrap Interface, the "Bootstrap-Write" MAY target just an Object ID, which will allow a Bootstrap-Server in using a TLV, SenML CBOR or SenML 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, SenML CBOR or SenML 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

Table: 6.1.7.5.-1 "Bootstrap-Write" Parameters

6.1.7.6. Bootstrap-Delete Operation

The "Bootstrap-Delete" operation targets one or several Object Instances and can be sent multiple times.

Only in the Bootstrap Interface, the "Bootstrap-Delete" operation MAY target any Instance or all Instances of any Object including the Security Object (ID:0), supported by the LwM2M Client. The two exceptions are the LwM2M Bootstrap-Server Account, potentially including an associated Instance of an OSCORE Object ID:21, and the single Instance of the mandatory Device Object (ID:3), which are not affected by any Delete operation.

When the "Bootstrap-Delete" operation is used without any parameter (i.e. without Object ID parameter), all Instances of all Objects in the LwM2M Client MUST be removed (except for the two cases mentioned above). This functionality could be used for initialization purposes before LwM2M Bootstrap-Server sends Write operation(s) to the LwM2M Client.

The "Bootstrap-Delete" operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	No	-	Indicates the Object from which Object Instance will be deleted. If no Object ID is indicated, all existing Object Instances (except the LwM2M Bootstrap-Server Account, potentially including an associated Instance of an OSCORE Object, and the Instance of the Device Object) in the LwM2M Client will be deleted.
Object Instance ID	No	-	Indicates the Object Instance to delete. If no Object Instance ID is indicated, then all Instances of the designated Object (Object ID MUST be provided) are deleted.

Table: 6.1.7.6.-1 Bootstrap-Delete Parameters

6.2. Client Registration Interface

The LwM2M Server MUST support all operations in this interface. The LwM2M Client MUST support the "Register" and the "Update" operations. The LwM2M Client SHOULD support the "De-register" operation.

The Client Registration Interface 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 7. Identifiers and Resources. When registering, the LwM2M Client performs the "Register" operation and provides the information required by the LwM2M Server (e.g. the supported Objects and existing Object Instances) as well as optional parameters (e.g. Endpoint Client Name). The LwM2M Client maintains the registration and communications session(s) with each LwM2M Server based upon the configured parameters (e.g. Lifetime, Queue Mode). The LwM2M Client periodically performs an update of its registration information to the registered LwM2M Server(s) by performing the "Update" operation.

If the lifetime of a registration expires without receiving an update from the LwM2M Client the LwM2M Server will consider it a de-registration:

- The LwM2M Server MUST remove the registration of that LwM2M Client and existing observations. If the LwM2M Client is unaware of the expiration, when the LwM2M Client performs a registration update the LwM2M Server will respond with an error.
- Upon receipt of the error message, the LwM2M Client SHOULD reset its state and register again. 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.

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.

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 about 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.

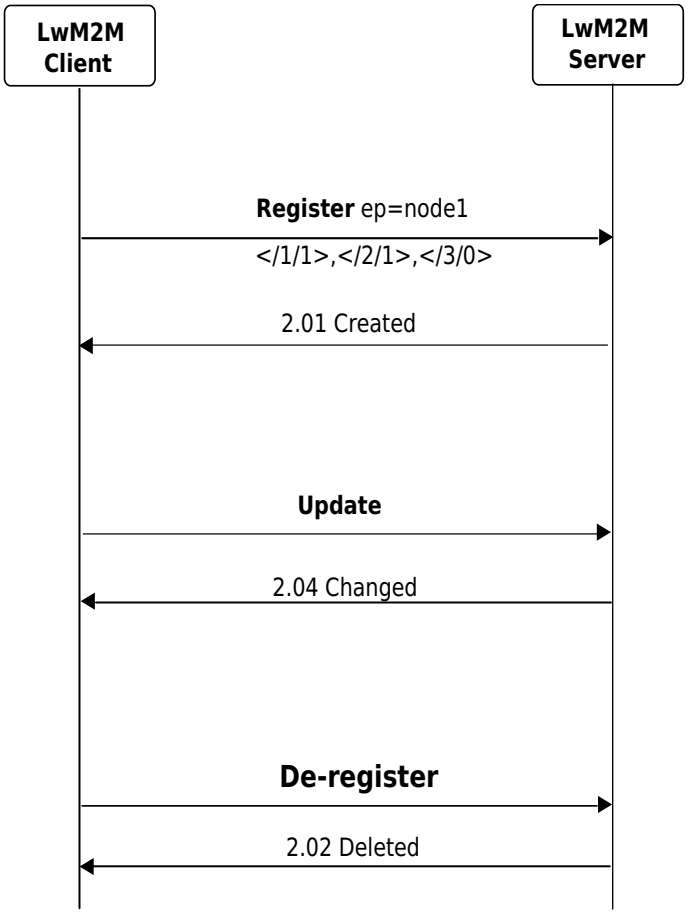


Figure: 6.2.-1 Client Registration Interface example flows

The mapping to underlying transports is detailed in [LwM2M-TRANSPORT].

6.2.1. Register Operation

Registration is performed when a LwM2M Client sends a "Register" operation to the LwM2M Server. After the LwM2M Device is turned on and the bootstrap procedure has been completed, the LwM2M Client MUST perform a "Register" operation to each LwM2M Server that the LwM2M Client has a Server Object Instance. Table: 6.2.1.-1 Registration

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

The "Register" operation SHOULD include the Endpoint Client Name parameter along with other parameters listed in Table: 6.2.1.-1 Registration Parameters. When the Endpoint Client Name parameter is provided in the "Register" operation then it MUST contain 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: 6.2.1.-1 Registration Parameters and the LwM2M Client MUST support LwM2M Version, Lifetime, Object and Object Instances and MAY support Binding Mode and SMS Number. With LwM2M v1.1, the Endpoint Client Name becomes an optional parameter for the LwM2M Client, but MUST be supported by the LwM2M Server.

Parameter	Required	Default Value	Notes
Endpoint Client Name	No		See Section 7.3. Identifiers
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. The LwM2M version number reported MUST correspond to the approved version number of this document.
Binding Mode	No	U	Indicates the supported binding modes in the LwM2M Client. This value SHOULD be the same as the value in the “Supported Binding and Modes” resource in the Device Object (/3/0/16). The valid values of the parameter are listed in Section 6.2.1.2. Behaviour with Current Transport Binding and Modes
Queue Mode	No		Indicates whether Queue Mode is supported. The Queue Mode is useful when the LwM2M Device is not reachable by the LwM2M Server at all times and it helps the LwM2M Client to reduce power consumption by sleeping longer.
SMS Number	No		The value of this parameter is the MSISDN or External Identifier where the LwM2M Client can be reached for use with the SMS binding or when both the LwM2M Client and the LwM2M Server support the SMS registration update trigger mechanism defined in [LwM2M-TRANSPORT].
Objects and Object Instances	Yes		The list of Objects supported and Object Instances available on the LwM2M Client (Security Object ID:0, and OSCORE Object ID:21, if present, MUST NOT be part of this list).

Table: 6.2.1.-1 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.

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, and OSCORE Object ID:21, if present, 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 and Object Version CoRE link parameter "ver" if required as it is defined in Section 7.2. Object Versioning of this document. Any other parameters included in the link MUST be silently ignored, unless specified for use by the LwM2M Enabler. Any optional resources not supported by the Server MUST be silently ignored.

The payload for a LwM2M Client supporting LwM2M Server, Access Control, Device, Connectivity Monitoring and Firmware Update Objects from Section Appendix E. LwM2M Objects defined by OMA (Normative) 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 Section Appendix F. Example LwM2M Client (Informative)):

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

If the LwM2M Client supports optional data formats, it MAY 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 110 is the value assigned in the CoAP Content-Format Registry for the SenML JSON format used by LwM2M).

</>;ct=110, </1/0>,</1/1>,</2/0>,</2/1>,</2/2>,</2/3>,</2/4>,</3/0>,</4/0>,</5>

6.2.1.1. Bootstrap and LwM2M Server Registration Mechanisms

In order to provide robust mechanisms for LwM2M Server registrations, resources have been defined in the LwM2M Server Object to configure the order of LwM2M Server registrations, registration retry behavior and the ability for the LwM2M Client to react during registration failure conditions. These resources are optional, so the behavior is only enabled when these resources are defined. If these resources are not defined, default values in Table: 6.2.1.1.-1 Registration Procedures Default Values or a manufacturer defined implementation may be used.

The first step is for the LwM2M Client to determine the registration order when multiple LwM2M Servers are defined (i.e. multiple instances of LwM2M Server Objects). The LwM2M Client orders the LwM2M Server registrations in ascending order of the "Registration Priority Order" value. It is recommended that each ordered LwM2M Server has a unique priority, so it is implementation dependent how to order those LwM2M Servers with the same priority. If this value is not defined for a LwM2M Server, registration attempts to that LwM2M Server are managed independently and do not impact other LwM2M Server registrations.

For each LwM2M Server that is defined in the registration order, the "Registration Failure Block" resource indicates the behavior of the LwM2M Client upon registration failure to that LwM2M Server. When the "Registration Failure Block" value is set to true and registration to the LwM2M server fails, the LwM2M Client performs additional registration retry sequences and blocks registration to other LwM2M Servers in the priority order. When the "Registration Failure Block"

value is set to false, the LwM2M Client proceeds with registration to the next LwM2M Server in the priority order whether the current LwM2M Server is successfully registered or not.

The procedure to manage the registration to LwM2M Servers that have a "Registration Priority Order" resource included is defined here:

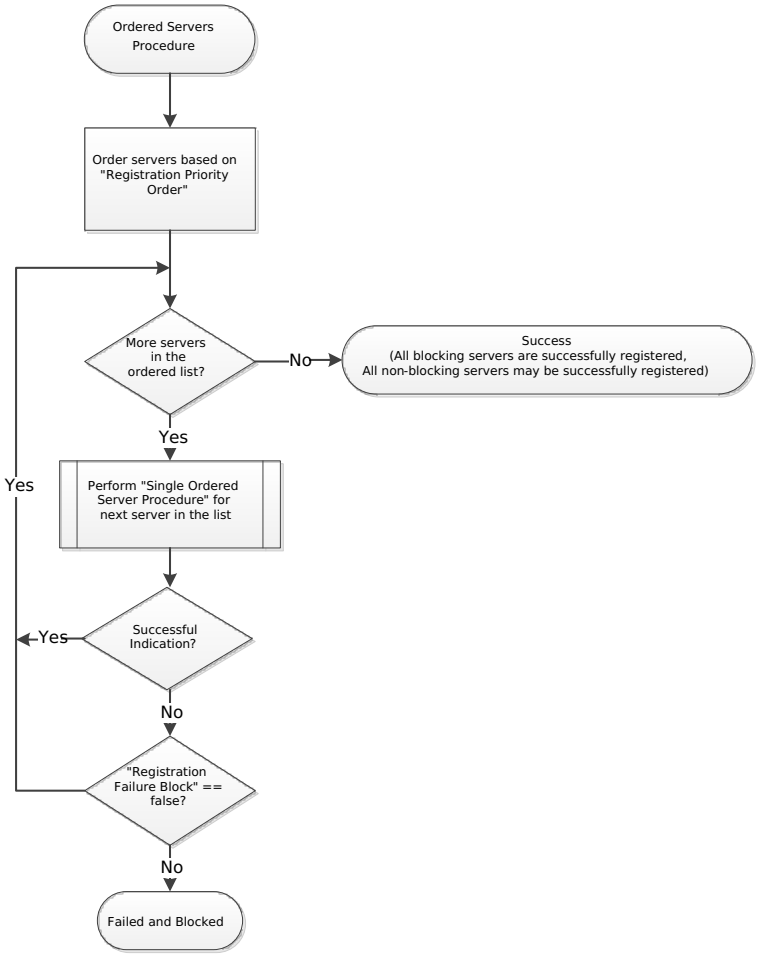


Figure: 6.2.1.1.-1 Ordered Servers Procedure

When registration fails to non-blocking LwM2M Servers in the "Ordered Servers Procedure", it is not specified how the LwM2M Client retries registration with those LwM2M Servers. One solution could be to execute the "Single Unordered Servers Procedure" for all failed non-blocking LwM2M Servers.

The "Communication Retry Count", "Communication Retry Timer", "Communication Sequence Delay Timer" and "Communication Sequence Retry Count" resources indicate to the LwM2M Client the retry mechanism to be used for

registration attempts to a single LwM2M Server. The registration mechanism to a LwM2M Server consists of a sequence of "Communication Retry Count" attempts within a retry sequence. The "Communication Retry Timer" value is multiplied by two to the power of the current attempt minus one (2^(Current Attempt-1)) to create an exponential back-off between attempts within a single retry sequence to a LwM2M Server. If one retry sequence of attempts to a LwM2M Server fails, then a new sequence of attempts may be retried after "Communication Sequence Delay Timer." The maximum number of retry sequences to a LwM2M Server is set by the "Communication Sequence Retry Count."

How these values are applied to those LwM2M Servers that are part of the ordered registration is defined in the "Single Ordered Server Procedure" here:

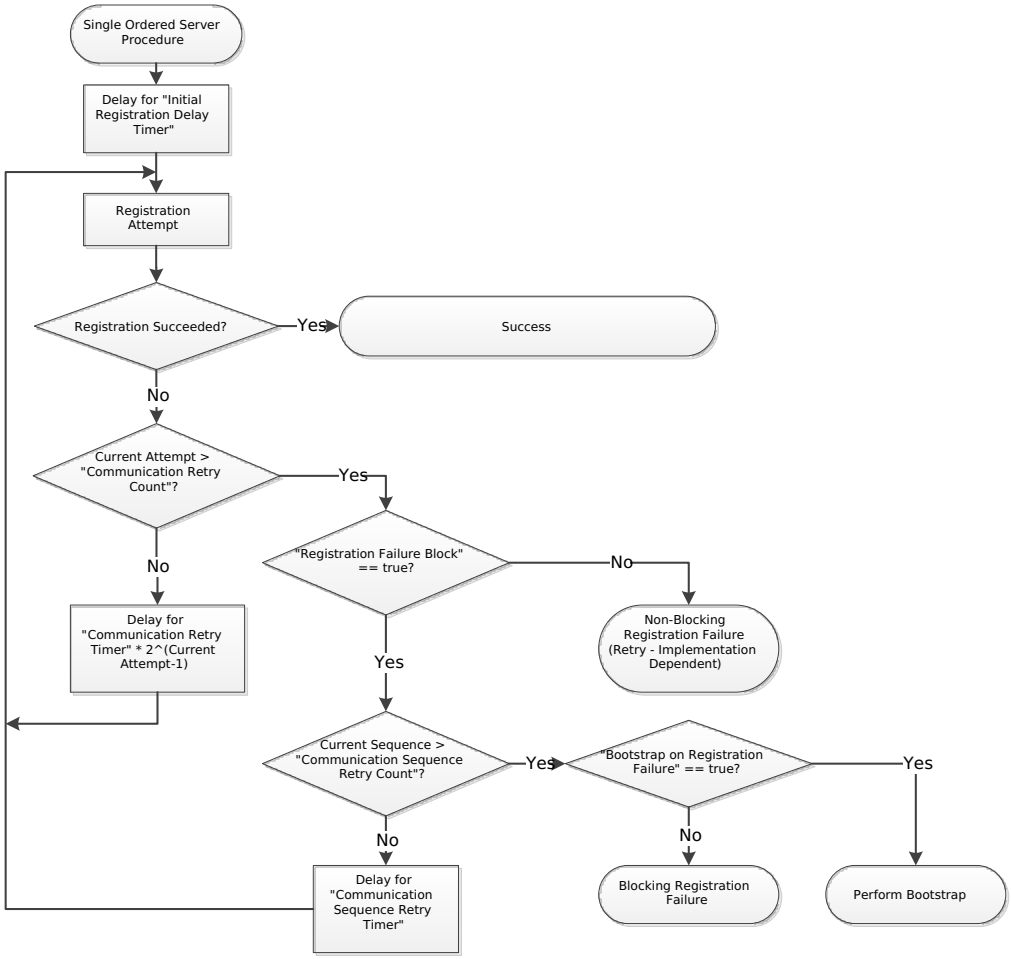


Figure: 6.2.1.1.-2 Single Ordered Server Procedure

How these values are applied to those LwM2M Servers that are NOT part of the ordered registration is defined in the "Single Unordered Server Procedure" here:

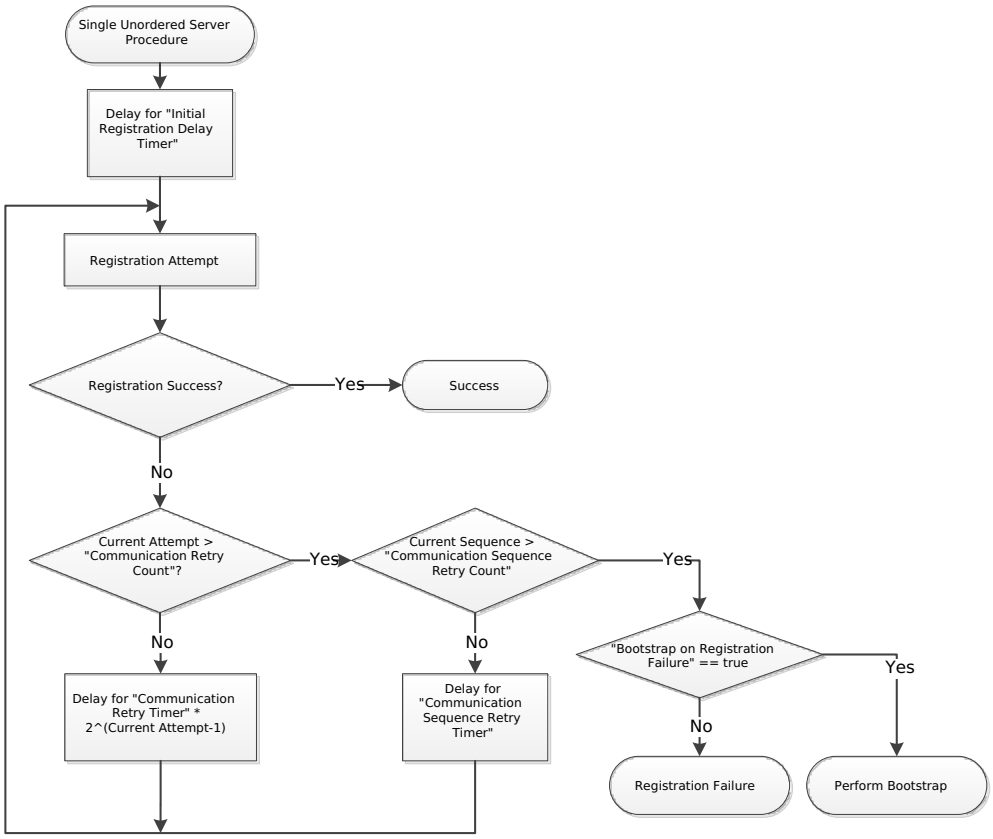


Figure: 6.2.1.1.-3 Single Unordered Server Procedure

Note: Current Attempt and Current Sequence start at 1 to represent the first attempt or sequence in the "Single Ordered Server Procedure" and "Single Unordered Server Procedure". These values should be reset to 1 at the start of each sequence and between sequences.

The "Initial Registration Delay Timer" defines the delay before the single registration procedure (ordered or unordered) is attempted for a LwM2M Server.

The "Bootstrap on Registration Failure" resource indicates to the LwM2M Client whether to re-bootstrap in case of a registration failure to the LwM2M Server. The exact behavior in case of bootstrap registration success, triggered by a server registration failure, is implementation dependent. For example, already successful LwM2M Server registrations may be maintained.

Note: It is recommended to set the "Bootstrap on Registration Failure" resource to true only for critical LwM2M Servers.

The registration retry mechanisms for the Bootstrap Server are implementation dependent.

The following table represents the recommended default values to use when the resources are not defined in the LwM2M Server Object:

Resource	Default Value
Registration Priority Order	Not defined
Registration Failure Block	False
Initial Registration Delay Timer	0 (seconds)
Bootstrap on Registration Failure	True
Communication Retry Count	5
Communication Retry Timer	1 (minute)
Communication Sequence Delay Timer	24 (hours)
Communication Sequence Retry Count	1

Table: 6.2.1.1.-1 Registration Procedures Default Values

6.2.1.2. Behaviour with Current Transport Binding and Modes

Behaviour of the LwM2M Server and the LwM2M Client is differentiated by Current Transport Binding and Modes. Current Transport Bindings are decided by "Binding" Resource set by the LwM2M Server. The "Binding" resource contains a combination of supported transports, for example UDP and TCP.

Table: 6.2.1.2.-1 Behaviour with Current Transport Bindings describes the behaviour of the LwM2M Server and the LwM2M Client for each Current Transport Bindings.

Current Transport Bindings	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.
T (TCP)	The LwM2M Server expects that the LwM2M Client is reachable via the TCP binding at any time. The LwM2M Server MUST send requests to a LwM2M Client using the TCP binding. The LwM2M Client MUST send the response to such a request over the TCP 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.

N (Non-IP)	Non-IP is defined in accordance with 3GPP TS 23.401 and is applicable for both NB-IoT and LTE M networks. Appendix C in [LwM2M-TRANSPORT] : LWM2M over 3GPP LPWA Networks - NB-IoT and LTE M, describes in detail the alternate routes for delivering non-IP data. Non-IP can also refer to alternate transports such as LoRAWAN The LwM2M Server MUST send requests to a LwM2M Client using the Non-IP binding. The LwM2M Client MUST send the response to such a request over the Non-IP binding.

Table: 6.2.1.2.-1 Behaviour with Current Transport Bindings

- While multiple transports are supported, only one transport binding can be used during the entire session. As an example, when UDP and SMS are both supported, the LwM2M Client and Server can choose to communicate either over UDP or SMS during the entire session.
- "Preferred Transport" is an optional resource in the LwM2M Server Object (/1/x/22). If this resource is defined, the client SHALL use this to initiate a connection over the specified transport, unless unable to do so. If this resource is not defined, the client SHALL use a client-preferred binding supported by both the server (from the list in the "binding" resource of the LwM2M server object - /1/x/7) and client (from the list in "Supported Binding and Modes" resource in the Device object - /3/x/16). The "Preferred Transport" resource can also be used to select a session specific required transport. For example, when Non-IP is normally indicated as preferred, this resource can be set to UDP temporarily to perform a firmware update.
- "Registration Update Trigger" is a mandatory resource in the LwM2M Server Object (/1/x/8). The client SHALL use the argument when the "Registration Update Trigger" resource is executed unless not supported by the client or not indicated as supported by the server in the "binding" resource of the LwM2M Server Object (/1/x/7). When that argument indicates an overriding binding that is supported by the client, the client SHALL immediately release the existing transport connection and establish a new transport connection using the overriding binding if that overriding binding is different than the current transport connection.
- The client SHALL assume that the server supports the UDP binding even if the server does not include UDP (“U”) in the "binding" resource of the LwM2M server object (/1/x/7).
- When the APN Connection Profile object is defined with a "PDN Type" resource (/11/x/24), that "PDN Type" SHALL be preferred when present in the "binding" resource of the LwM2M server object (/1/x/7).
- When configurations related to the bindings change, those new values SHALL NOT affect the current communication session, and the new values SHALL be applied to the future sessions. The existing LwM2M client registration SHALL NOT be affected by the use of updated bindings configuration.
- Queue Mode and device triggering can be enabled independently, and can be used in conjunction with one or more transports. Using the device triggering, the LwM2M Server MAY request the LwM2M Client to perform operations, such as the "Update" operation by sending an "Execute" to the "Registration Update Trigger" Resource via SMS. A LwM2M Client is not expected to send an SMS response for a device trigger message.

6.2.2. Update Operation

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: 6.2.2.-1 Update Parameters changes, the LwM2M Client MUST send an "Update" operation to the LwM2M Server. This "Update" operation MUST contain only the parameters listed in Table: 6.2.2.-1 Update Parameters 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: 6.2.2.-1 Update Parameters

When present,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, and the OSCORE Object ID:21, if present, MUST NOT be part of Update Objects and Object Instances list.

The payload Media-Type of that "Update" operation is the same as the one of the "Registration" operation.

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

Two common operations are:

1. Extending the lifetime of a registration

In this case the LwM2M Client sends an "Update" operation with no parameters.

Figure: 6.2.2.-1 Update Example Flow #1 shows an example exchange where the LwM2M Client sends an "Update" operation that only refreshes the registration, i.e. the message does not contain any parameters. With the second "Update" the Client changes the lifetime field to 6000 (seconds) and hence the lt parameter is included in the message.

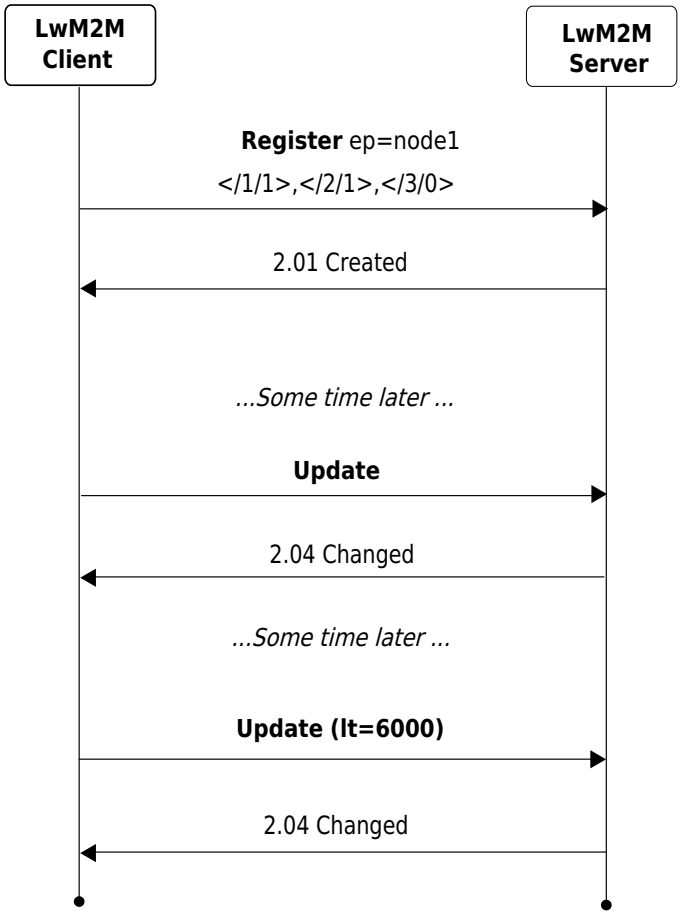


Figure: 6.2.2.-1 Update Example Flow #1

2. Adding and removing Objects and Object Instances

In this case the LwM2M Client sends an "Update" with a body listing the complete list of objects and object instances.

Figure: 6.2.2.-2 Update Example Flow #2 shows an example exchange whereby the LwM2M Client starts with an initial registration of two instances for a LwM2M Object with ID 12. Later, the LwM2M Server adds a third instance and subsequently deletes the second. With the "Update" operation the LwM2M Client includes the new list of Objects and Object Instances.

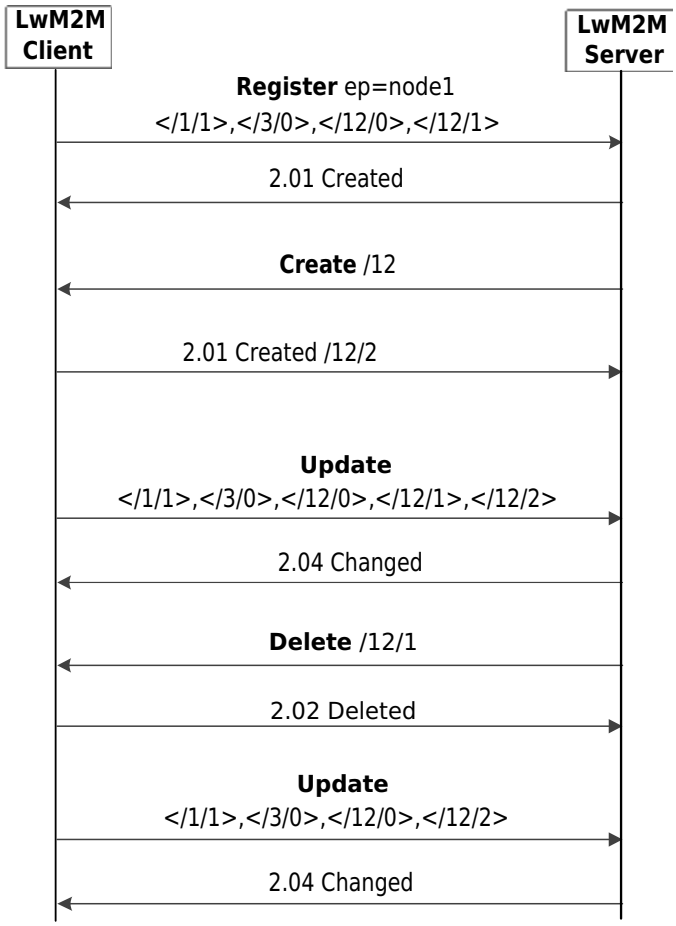


Figure: 6.2.2.-2 Update Example Flow #2

6.2.3. De-register Operation

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.

6.3. Device Management and Service Enablement Interface

The LwM2M Server and the LwM2M Client MUST support all the operations on this interface unless clearly stated otherwise. Support for some of the non-mandatory operations may also be dependent on the choice of transport layer and the services it provides.

The Device Management and Service Enablement Interface 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", "Read-Composite", "Write", "Write-Composite", "Delete", "Execute", "Write-Attributes", or "Discover" operations. The operations that a Resource supports are defined in the Object definition using the Object Template. The Object Template is described in Section Appendix D. LwM2M Object Template and Guidelines (Normative). The Normative Objects defined by the LwM2M Enabler are described in Section Appendix E. LwM2M Objects defined by OMA

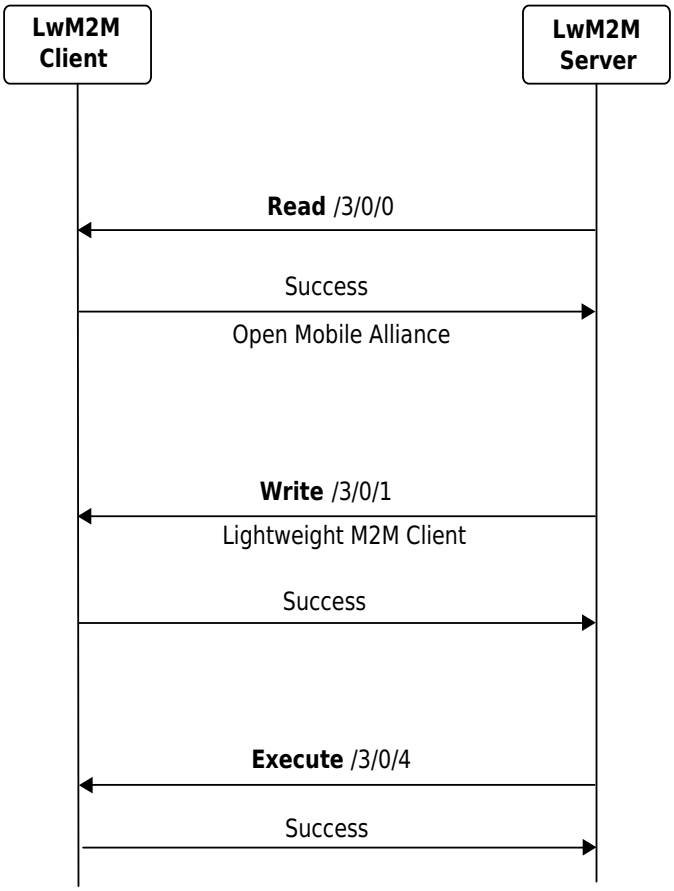
(Normative).

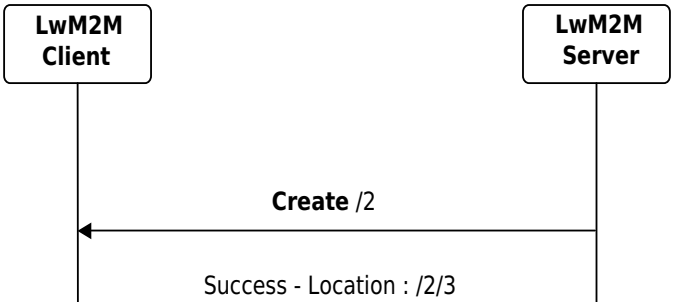
The LwM2M Client MUST ignore LwM2M Server operations on this interface until the registration procedure with the respective LwM2M Server is completed.

The LwM2M Client MUST reject any LwM2M Server operation on the Security Object (ID: 0) with an "4.01 Unauthorized" response code.

The LwM2M Client MUST reject any LwM2M Server operation on an OSCORE Object (ID: 21) with an "4.01 Unauthorized" response code.

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





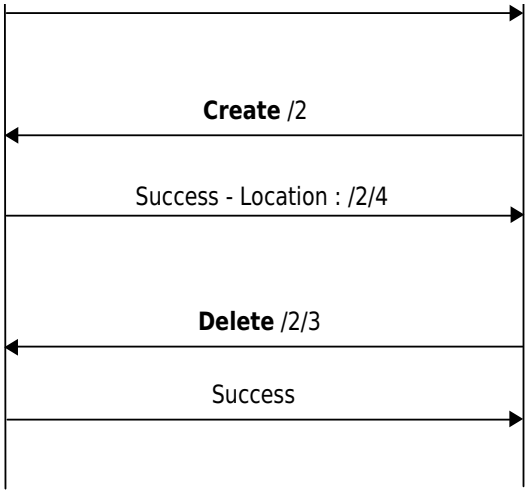


Figure: 6.3.-1 Example flows of Device Management & Service Enablement Interface

The mapping to underlying transports is detailed in [LwM2M-TRANSPORT].

6.3.1. Read Operation

The "Read" operation is used to access the value of a Resource, a Resource Instance, 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 Resource ID and Resource Instance ID MUST NOT be indicated otherwise an error is returned by the Client; if no error, the authorized Instances of the Object are returned to the Server
Resource ID	No	-	Indicates the Resource to read. If the final target is not a Resource Instance (see below), then the Resource value is returned. If no Resource ID is indicated, then the Resource Instance ID MUST NOT be indicated as well, and the whole Object Instance is returned
Resource Instance ID	No	-	Indicates the Resource Instance to read. An error MUST be returned by the Client, if the Resource ID is absent or if the targeted Resource is not defined as a Multiple-Instance Resource.

Table: 6.3.1.-1 Read Parameters

Any optional resources included in the "Read" operation response that are not supported by the LwM2M Server MUST be silently ignored.

6.3.2. Discover Operation

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 assigned or attached Attributes including the Object Version attribute if required (see Section 7.2. Object Versioning and Table: 5.1.2.-1 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: 6.3.2.-1 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 been assigned to this Object level, are also returned, see Section 5.1.2. Attributes Classification.

  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>,</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-Attribute assigned to the Object level.

- If Object ID and Object Instance ID are only specified, the list of Attributes assigned to that Object Instance MUST be reported, and the list of instantiated Resources and their assigned Attributes 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=2,</3/0/7>;dim=2;gt=50;lt=42.2, </3/0/8>;dim=2,</3/0/11>,</3/0/16>

  means that regarding the Device Info Object Instance, an R-Attribute has been assigned to this Instance level; the LwM2M Client supports the Multiple-Instance Resources 6, 7, and 8 with a dimension of 2 and has 2 additional Notification parameters assigned to Resource 7.

- If Object ID, Object Instance ID and Resource ID are specified, the attached Attributes of that Resource MUST be returned (includes the assigned R-Attributes and the R-Attributes inherited from the Object and Object Instance) as well as the list of the Resource Instances in case of a Multiple-Instance Resource (includes the assigned R-Attribute(s) if any).

  For example: if Object ID is 3, Object Instance ID is 0, and Resource ID is 7, then

  </3/0/7>;dim=2;pmin=10;pmax=60;gt=50;lt=42.2, </3/0/7/0>, </3/0/7/1>;lt=45

  with pmin assigned at the Object level, pmax assigned at the Object Instance level, and "lt" Attribute explicitly set

at a Resource Instance level (/3/0/7/1)..

- If a Resource, an Object Instance, or an Object have attributes for multiple LwM2M Servers, then the Attributes returned in the link, MUST only be related to the Server which performed the "Discover" operation. 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 the "Discover" operation on both Servers will be differentiated e.g.

  from Server 1: Discover /3/0/7 could provide the answer: </3/0/7>;dim=2;gt=50;lt=42.2, </3/0/7/0>, </3/0/7/1>

  from Server 2: Discover /3/0/7 could provide the answer: </3/0/7>;dim=2;pmax=300;gt=80;lt=65.5, </3/0/7/0>, </3/0/7/1>

Any optional resources included in the "Discover" operation response that are not supported by the LwM2M Server MUST be silently ignored.

6.3.3. Write Operation

The "Write" operation is used to change the value of a Resource, the value of a Resource Instance, the values of an array of Resources Instances or the values of multiple Resources from an Object Instance. The "Write" operation can also be used to request the deletion or the allocation of specific Instances of a Multiple-Instance Resource

The request includes the value to be written encoded in one of the data format defined in Section 7.4. Data Formats for Transferring Resource Information.

The TLV, SenML CBOR or SenML JSON data format MUST be used in a "Write" request:

- when more than a single value of a Resource has to be changed
- when one or several Instances of a Multiple-Instance Resource have to be deleted or allocated.

The Write request MUST be rejected:

- if the specified data format is not supported by the LwM2M Client
- or if, checking the value of the incoming Resource against its data type, at least one of the following conditions is not met:
  - the value matches the expected format
  - the value is in the range specified in the Object definition
  - if the value is an Objlnk value, it must either point to an Object actually present in the LwM2M Client (the value will then be ObjectID:MAX_ID), or point to an Object Instance actually present in the LwM2M Client, or contain the null pointer (the value will then be MAX_ID:MAX_ID)
- or if the deletion or allocation of an Instance of a Multiple-Instance Resource is not allowed by the LwM2M Client.

The 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. When the Resource is a Multiple-Instance Resource, the existing array of Resource Instances is replaced to the condition the LwM2M Client authorizes that operation.

- Partial Update: updates Resources provided in the new value and leaves other existing Resources unchanged. When the Resource is a Multiple-Instance Resource, the existing array of Resource Instances is updated meaning some Instances may be created or overwritten to the condition the LwM2M Client authorizes such operations. Deleting via Partial Update is not possible.

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. If the final target is not a Resource Instance (see below), the payload is the new value for the Resource.The payload is the new value for the Resource. If no Resource ID is indicated, the Resource Instance ID MUST NOT be indicated , then the value included payload is an Object Instance containing the Resource values.
Resource Instance ID	No	-	Indicates the Resource Instance to write. An error MUST be returned by the Client, if the Resource ID is absent, if the targeted Resource is not defined as a Multiple-Instance Resource, or if the targeted Resource Instance doesn't exist.
New Value	Yes	-	The new value included in the payload to update the Object Instance, a Resource or a Resource Instance.

Table: 6.3.3.-1 Write Parameters

Any optional resources included in the "Write" operation that are not supported by the LwM2M Client MUST be silently ignored.

6.3.4. Write-Attributes Operation

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. Attributes Definitions and Rules fully apply here. Table: 5.1.2.-1 Class Attributes in Section 5.1.2. Attributes Classification provides explanation on the Attributes supported by the "Write-Attributes" operation: Minimum Period, Maximum Period, Greater Than, Less Than, Step, Minimum Evaluation Period and Maximum Evaluation Period.

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

Including <NOTIFICATION> class Attributes specified in Table: 5.1.2.-1 Class Attributes Section 6.2.1. Register Operation, the "Write-Attributes" operation has the following parameters:

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 and Resource Instance ID parameters MUST NOT be indicated and Attributes in the parameters of the operation are valid for all resources of all instances of the Object according to the precedence rules (Section 5.1.1. Attributes Definitions and Rules).

Resource ID	No	-	Indicates the Resource. When this parameter is omitted, then the ResourceInstance ID MUST NOT be indicated as well and it means the Attributes of this operation are set at an upper level (Object or Object Instance level).
Resource Instance ID	No	-	Indicates the Resource Instance. The LwM2M Client MUST return an error if the Resource ID is absent or if the targeted Resource is not defined as a Multiple-Instance Resource.
<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 operation (Object, Object Instance, Resource, or Resource Instance levels).

Table: 6.3.4.-1 Write-Attributes Parameters

6.3.5. Execute Operation

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).

The "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 "Execute" operation accepts arguments. Arguments MUST be in plain text, SenML CBOR or SenML JSON format. If the arguments are expressed in plain text format then they MUST follow the ABNF syntax below.

Table: 6.3.5.-1 Execute parameters

For the plain text format the syntax of the arguments is given in ABNF syntax 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:

1. 5
2. 2='10.3'
3. 7, 0=' https://www.omaspecworks.org'
4. 0,1,2,3,4

6.3.6. Create Operation

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 8. Access Control and its sub-sections. If any error occurs, it MUST NOT create anything.

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: 6.3.6.-1 Create parameters

The new value included in the payload MUST follow the TLV, SenML CBOR or SenML JSON data format according to Section 7.4. Data Formats for Transferring Resource Information.

The LwM2M Client MUST silently 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/SenML CBOR/SenML JSON payload of the "Create" operation, 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 (see Section 8. Access Control)

- Access Control Owner MUST be the LwM2M Server
- The LwM2M Server MUST have full access rights

6.3.7. Delete Operation

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 only exception concerns the single Instance of the mandatory Device Object (ID:3) which SHALL NOT be affected by any Delete operation.

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: 6.3.7.-1 Delete parameters

6.3.8. Read-Composite Operation

The LwM2M Client MAY support the "Read-Composite" operation.

The "Read-Composite" operation can be used by the LwM2M Server to selectively read any combination of Objects, Object Instance(s), Resources, and/or Resource Instances of different or same Objects in a single request. The list of elements to be read are provided as SenML Pack where the records contain Base Name and/or Name Fields, but no Value fields. The Read-Composite operation is treated as non-atomic and handled as best effort by the client. That is, if any of the requested resources do not have a valid value to return, they will not be included in the response. Section 7.4.5. SenML JSON shows examples of Read-Composite use.

6.3.9. Write-Composite Operation

The LwM2M Client MAY support the "Write-Composite" operation.

In contrast to "Write" operation, the scope of which is limited to a Resource(s) of a single Instance of a single Object, the "Write-Composite" operation can be used by the Server to update values of a number of different Resources across different Instances of one or more Objects. Similar to Read-Composite the Write-Composite operation provides a list of all resources to be updated, and their new values, using the SenML JSON/CBOR format. Unlike for Write operation, the Resources that are not provided are not impacted by the operation. Examples are shown in Section 7.4.5. SenML JSON.

The "Write-Composite" operation is atomic and cannot have partial success. That is, if the client supports this operation, it MUST reject a Server request where it cannot successfully write all the requested values to the requested list of Resources. Therefore, before processing Write-Composite, the client MUST ensure that all addressed objects exist and that the Server has write access to those Objects and Resources.

6.4. Information Reporting Interface

The Information Reporting Interface 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" or "Observe-Composite" operation to the L2M2M Client for an Object, an Object Instance or a Resource. An observation ends when a "Cancel Observation" or "Cancel Observation-Composite" operation is performed.

The LwM2M Server and the LwM2M Client MUST support all the operations on this interface unless clearly stated otherwise.

The LwM2M Client MUST ignore LwM2M Server operations 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.

The LwM2M Client MUST reject with an "4.01 Unauthorized" response code any LwM2M Server operation on the Security Object (ID: 0).

The LwM2M Client MUST reject with an "4.01 Unauthorized" response code any LwM2M Server operation on an OSCORE Object (ID: 21).

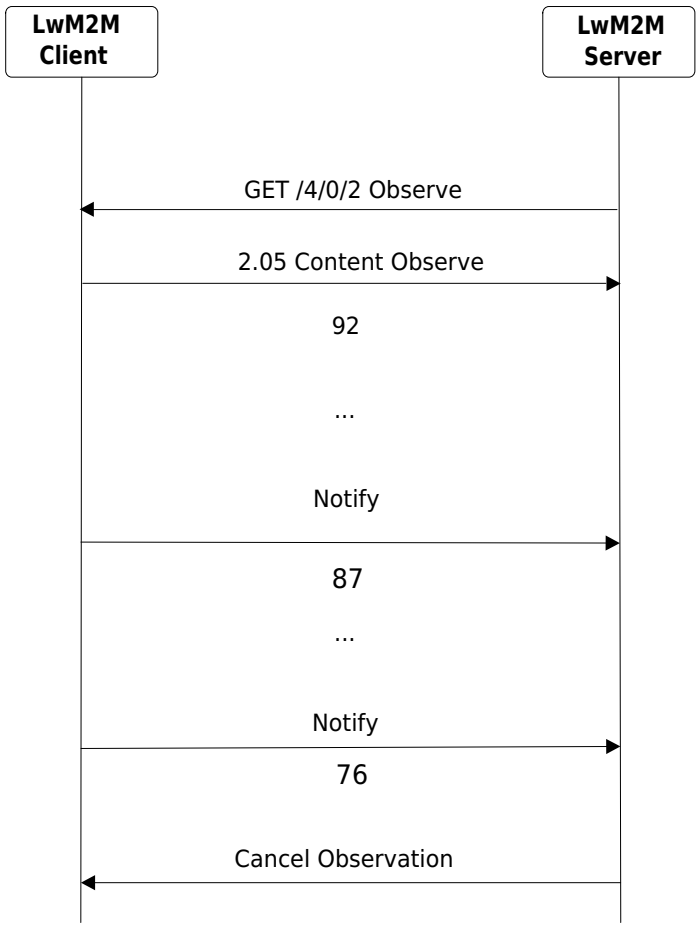


Figure: 6.4.-1 Example flow for Information Reporting Interface for the RSSI Resource of the Connectivity Monitoring Object of the example client
Appendix LwM2M Objects defined by OMA (Normative)

The mapping to underlying transports is detailed in [LwM2M-TRANSPORT].

6.4.1. Observe Operation

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 Section 6.3.4. Write-Attributes Operation 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 Resource ID and Resource Instance ID MUST NOT be indicated as well, otherwise an error is returned by the Client; if no error, all the Instances of the Object are observed.
Resource ID	No	-	Indicates the Resource to observe. If no Resource ID is indicated, then the Resource Instance ID MUST NOT indicated as well, and the whole Object Instance is observed.
Resource Instance ID	No	-	Indicates the Resource Instance to observe. An error MUST be returned by the Client, if no Resource ID is indicated or if the targeted Resource is not defined as a Multiple-Instance Resource.

Table: 6.4.1.-1 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.

6.4.2. Notify Operation

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 MUST 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: 6.4.2.-1 Notify Parameters

The following example shows how the Minimum and Maximum period parameters work as shown in Section 6.3.4. Write-Attributes Operation. 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 attributes 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.

Note: In case an "Observe" operation is initiated on a Resource, an Object Instance or an Object without any conditions previously configured, and with the "Default Minimum Period" in the LwM2M Server Object set to 0 (possibly by default), the "Notify" operation will be sent upon any change of, respectively, the observed Resource value, any Resource value of the observed Object Instance or any Resource value of any Instance of the observed Object. As specified in Section 6.4.1. Observe Operation and in Table: 6.4.1.-1 Observe Parameters, in case of an "Observe" operation on a certain Object the "Notify" operation includes all Resources of all Instances of the observed Object.

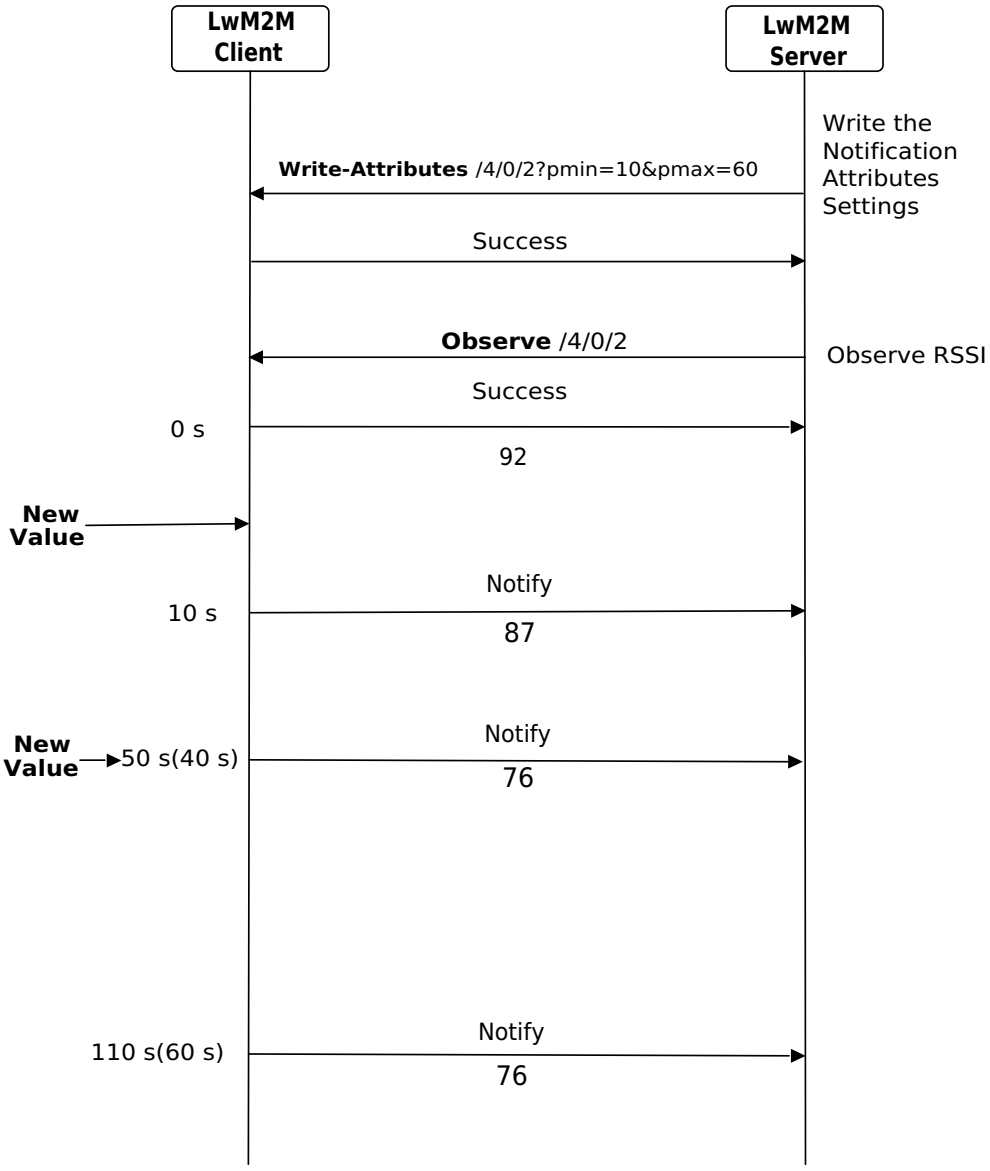


Figure: 6.4.2.-1 Example of Minimum and Maximum periods in an Observation

In this example 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.

6.4.3. Cancel Observation Operation

The "Cancel Observation" operation is sent from the LwM2M Server to the LwM2M Client to end an observation relationship that was previously created with an "Observe" operation.

6.4.4. Observe-Composite Operation

The LwM2M Client MAY support the "Observe-Composite" operation.

The LwM2M Server can use the "Observe-Composite" operation to initiate observations for a group of resources and/or resource instances across multiple object instances within the client. As with the "Read-Composite" operation, the list of elements to be observed is provided as a separate parameter to the operation in SenML JSON/CBOR format.

The <NOTIFICATION> Class Attributes for each resource that is observed through the "Observe-Composite" operation are individually configured by the LwM2M server using the "Write-Attribute". Each resource can have multiple observe conditions attached. If any of the conditions attached to one or more resources under observation meets the notification criteria a notify will be generated by the LwM2M client, which will include the value for each of the resources listed in the "Observe-Composite" operation. Hence, the resources that have not met the notify condition will still be included in the notification message.

Note: It is the responsiblity of the LwM2M Server to ensure that the Minimum and Maximum Period Attributes attached to observed resources follow the rules defined in Section 5.1.2. Attributes Classification. For example, the Minimum Period Attributes attached to targeted resources must be less than the Maximum Period Attributes attached to all targeted resources.

6.4.5. Cancel Observation-Composite Operation

If the "Observe-Composite" operation is supported by the client, the "Cancel Observation-Composite" operation MUST be supported by the Client.

The "Cancel Observation-Composite" operation is sent from the LwM2M Server to the LwM2M Client to end the previously set up composite observation relationship.

6.4.6. Send Operation

The LwM2M Client and the LwM2M Server MAY support the "Send" operation.

The "Send" operation is used by the LwM2M Client to send data to the LwM2M Server without explicit request by that Server.

The "Send" operation can be used by the LwM2M Client to report values for Resources and Resource Instances of LwM2M Object Instance(s) to the LwM2M Server. The Resources and Resource Instances to send is implementation specific. The LwM2M Server MAY use the "Mute Send" Resource in the LwM2M Server Object (Resource ID 23) to enable or disable the use of the "Send" operation. When information reporting should be controlled by the LwM2M Server, the "Observe" or "Observe-Composite" operations can be used.

The reported LwM2M Object Instances (potentially different Object types) MUST have been registered by the LwM2M Client to the LwM2M Server. That LwM2M Server MUST have Read access right granted on the Object Instances of the reportable Resources which MUST support the Read operation (see Section 8.2. Authorization). If these criteria are not

met for some Resources, the LwM2M Client MUST NOT include those Resources in the Send operation.

Any optional resources included in the "Send" operation that are not supported by the LwM2M Server MUST be silently ignored.

In addition, if any of the rules mentioned below is violated, the Server MUST respond with an error in reply to the "Send" operation.

The value included in the payload MUST be either SenML JSON or SenML CBOR format described in Section 7.4. Data Formats for Transferring Resource Information. As example:

Reporting the location of the LwM2M Client and the radio signal strength in a JSON payload:

```
Send
New Value:
[
 {"n":"/6/0/0", "v":43.61092},
 {"n":"/6/0/1", "v":3.87723},
 {"n":"/4/0/2", "v":-49}
]
```

7. Identifiers and Resources

This section defines the identifiers and resource model for the LwM2M Enabler.

7.1. 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. Each Resource is given a unique identifier within that Object and a data type among those defined in Section Appendix C. Data Types (Normative).

Figure: 7.1.-1 Relationship between LwM2M Client, Object, and Resources 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.

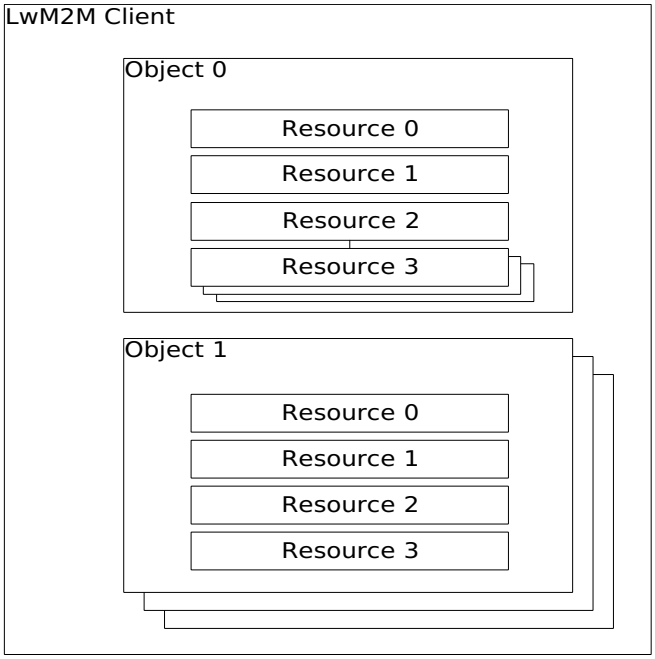


Figure: 7.1.-1 Relationship between LwM2M Client, Object, and Resources

Each Object, defined for the LwM2M Enabler, is assigned an unique OMA LwM2M Object identifier allocated and maintained by the OMA Naming Authority [OMNA]. The LwM2M Enabler defines standard Objects and Resources, see Section Appendix E. LwM2M Objects defined by OMA (Normative). 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 first be 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)

Data Type checking on a Resource value MUST be performed:

- by the Client when the Server tries to set such a Resource,
- by the Server when the Server retrieves such a Resource

The Client MUST reject a request, if an error is detected when checking an incoming Resource value against its data type.

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 operation 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-Instance Resources can be instantiated by LwM2M Server operations in using SenML JSON, SenML CBOR or TLV formats (see Section 7.4. Data Formats for Transferring Resource Information). The LwM2M Client also has the capability to instantiate Single or Multiple-Instance Resources.

The LwM2M Server performs operations on an Object, Object Instance and Resources as described in Section 6. Interfaces. These operations are conveyed as described in [LwM2M-TRANSPORT] and how to convey the Operation data is defined in Section 7.4. Data Formats for Transferring Resource Information.

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: 7.1.-2 Example of Supported operations and Associated Access Control Object Instance 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 8. Access Control.

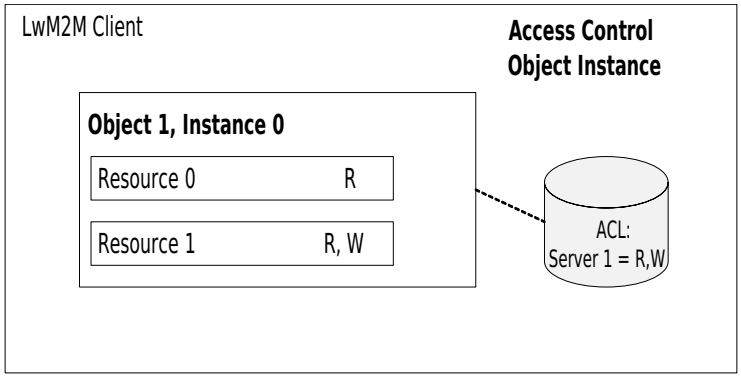


Figure: 7.1.-2 Example of Supported operations and Associated Access Control Object Instance

7.2. Object Versioning

7.2.1. 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", can 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 (i.e. schema) intended to be registered by the LwM2M Client:

- for LwM2M Objects specified within the LwM2M Enabler specification, and
- for LwM2M Objects specified outside of a LwM2M Enabler specification (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).

The official specification of the Objects is available at [OMNA].

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 mandatory Resource is added to or removed from the Object.
    - 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

7.2.2. 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 7.2.1. General Policy)
- 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: Section 7.2.1. General Policy)

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 7.3. Identifiers and Section Appendix J. LwM2M Schema and Object Definition File (Informative) for further information.

7.2.3. 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: 7.2.3.-1 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, the 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>

7.3. Identifiers

The LwM2M Enabler defines specific identifiers for entities used within the LwM2M Protocol. These identifiers are defined in Table: 7.3.-1 LwM2M Identifiers.

Identifier	Format	Description
Endpoint Client Name	String	Identifies the LwM2M Client on one LwM2M Server (including LwM2M Bootstrap-Server). This parameter is optional and provided to the LwM2M Server during Registration, also provided to LwM2M Bootstrap-Server when executing the Bootstrap procedure. It MUST be provided when the security protocol does not provide an authenticated identifier or this information is not available for the server. The Endpoint Client Name is a string. Recommended URI and URN formats are documented in Section [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. Assigned during the Bootstrap procedure. The values '0' and MAX_ID '65535' are reserved values and MUST NOT be used for identifying a 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 an Object in the LwM2M Client. This identifier is assigned by OMA. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying an Object.
Object Instance ID	16-bit unsigned integer	Uniquely identifies an Object Instance of an Object within the LwM2M Client. This identifier is assigned by the LwM2M Client or LwM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying an Object Instance.
Resource ID	16-bit unsigned integer	Uniquely identifies a Resource within an 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. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying a Resource.
Resource Instance ID	16-bit unsigned integer	Uniquely identifies a Resource Instance of a Resource. This identifier is assigned by a LwM2M Client or LwM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying a Resource Instance.

Table: 7.3.-1 LwM2M Identifiers

7.3.1. Endpoint Client Name

The Endpoint Client Name is a string. URIs and URNs are RECOMMENDED formats for this identifier. The Endpoint Client Name identifier MUST be unique at the server or servers it is being used with. Note that the Endpoint Client Name is unauthenticated and can be set to arbitrary value by a misconfigured or malicious client and hence MUST NOT be used alone for any decision making without prior matching the Endpoint Client Name against the identifier used with the security protocol protecting LwM2M communication. This parameter MAY be omitted when the security protocol provides to the server an authenticated identifier that is identical to this parameter. For more discussion consult the security consideration section in the [LwM2M-TRANSPORT] specification. This security consideration section also provides a description of the privacy implications of re-using the same identifiers across multiple servers.

This specification RECOMMENDS the Endpoint Client Name to be in a URI or a URN format for uniqueness reasons. The following table lists recommended formats. Other URI and URN formats MAY also be used. In particular, URN formats defined in [DMREPPRO] Section 6.4. Information Reporting Interface can be used. Note: Implementations need to be careful when using the Endpoint Client Name to perform database lookups due to the risk of SQL injection attacks.

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: ###############-############### IMEI-IMSI URN: Identify a device using a combination of International Mobile Equipment Identifier [3GPP-TS_23.003] and IMSI. IMEI is 15 digits and IMSI is 15 digits. The format of the URN is urn:imei-imsi: ###############-############### NAI URN: Identify a device using a combination of "Local Identifier@Domain Identifier" [3GPP-TS_23.003]. Further [3GPP-TS_23.003] references RFC 4282, which is replaced by [RFC7542] provides total limit to 63 octets (though allowed length is 253 octets which is not adviced in [RFC7542] as it may hinder different systems to accommodate that many octets). The format of the URN Is urn:extid:##. 63 octets. Examples are: 1. The NAI 123456789@domain.com is represented as urn:nai:123456789@domain.com 2. The NAI user@homerealm.example.net is represented as urn:nai:user@homerealm.example.net

Table: 7.3.1.-1 Endpoint Client Name

If a URN identifier from the above table is used, it MUST adhere to the specified format.

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

7.3.2. 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 a Reusable Resource and its usage when hosted in an Object specification MUST follow several rules:

- the Name, ID, Operation, Type, Range and Units are frozen characteristics when the Reusable Resource is registered at [OMNA] and cannot not be changed when such a Resource is specified in the definition of a hosting Object.
- the registered Description field of a Reusable Resource is also a frozen characteristic; when extension is required to fit the real need of the Object hosting such a Resource, that extension must be mentioned in the Object specification but outside of the Description field itself; furthermore any extension MUST be compatible with the content of the registered Description field.
- a Reusable Resource is registered without defining "Multiple-Resource", or "Mandatory" characteristics; both characteristics will be determined when such a Resource is specified in the definition of a hosting Object.

Note: The strict guidelines about reusable resources, as indicated in the above bullets, are specified to ensure that the definition remains consistent across different implementations. [OMNA] is the only official reference for reusable resources to which all implementations should adhere. If the existing reusable resources are not sufficient to meet the requirements of operators/vendors, it is recommended to propose a new reusable resource.

7.4. Data Formats for Transferring Resource Information

The following data formats are defined by the LwM2M Enabler in this section: plain text, opaque, TLV, JSON, CoRE Link, CBOR, SenML JSON, and SenML CBOR.

The LwM2M Server MUST support all data formats, including TLV that was mandatory to support for LwM2M TS 1.0 Clients.

The LwM2M Client MUST support plain text, opaque, and CoRE Link data formats and it SHOULD support at least one of SenML CBOR or SenML JSON data formats. In addition, a LwM2M Client MAY choose to support other data formats. LwM2M Clients supporting LwM2M TS 1.1 or newer SHOULD NOT use the legacy TLV or JSON formats.

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

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 Types 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 Format	application/link-format	40
Opaque	application/octet-stream	42
TLV	application/vnd.oma.lwm2m+tlv	11542

JSON	application/vnd.oma.lwm2m+json	11543

Table: 7.4.-1 IANA registered Media Types supported in LwM2M TS 1.0

In addition, the LwM2M TS 1.1 adds support for the following standardized media types.

Data Format	IANA Media Type	Numeric Content-Formats [CoAP]
CBOR	application/cbor	60
SenML JSON	application/senml+json	110
SenML CBOR	application/senml+cbor	112

Table: 7.4.-2 Additional standardized Media Types supported in LwM2M TS 1.1

7.4.1. 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 Section Appendix C. Data Types (Normative).

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.

7.4.2. 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.

7.4.3. CBOR

The Concise Binary Object Representation format is used for "Read" and "Write" operations on singular Resources.

This data format has a Media Type of application/cbor which covers all the LwM2M data types via the use of its major types with or without optional tags.

7.4.4. TLV

For "Read" and "Write" operations, the binary TLV (Type-Length-Value) format can be 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: obxxxxxxxx (MSB is the bit following ob) 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. Data Types (Normative)).

Table: 7.4.4.-1 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.

Figure: 7.4.4.-1 TLV nesting 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.

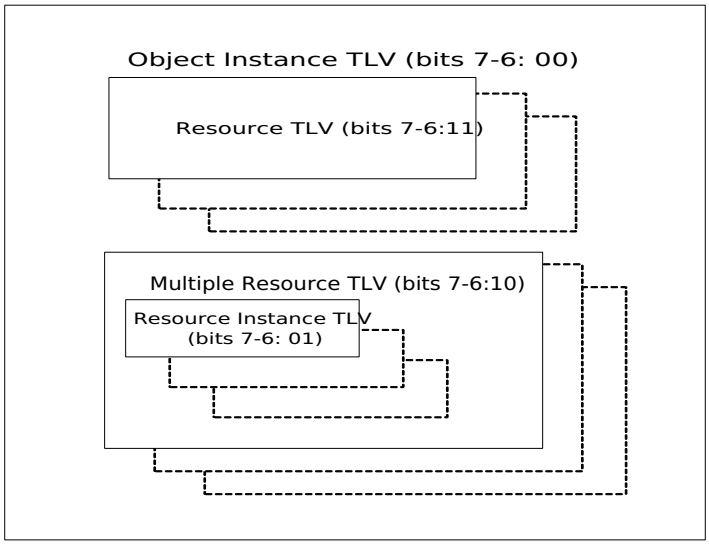


Figure: 7.4.4.-1 TLV nesting

7.4.4.1. 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

Table: 7.4.4.1.-1 Single Object Instance Request Example

7.4.4.2. 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

Table: 7.4.4.2.-1 Request on Single-Instance Object

B) Request on Multiple-Instance 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-Instance 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

Table: 7.4.4.2.-2 Request on Multiple-Instance Object having 2 instances

C) Request on Multiple-Instance 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

Table: 7.4.4.2.-3 Example, a request to the Server Object Instances of a LwM2M client is performed (Read /1)

7.4.4.3. Example of Request on an Object Instance containing an Object Link Resource

Examples are based on the LwM2M Object Tree illustration presented in Figure: C.-1 Object link Resource simple illustration. The TLV format doesn't report the 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

Table: 7.4.4.3.-1 Example 1) Request to Object 65 Instance 0: Read /65/0

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

Table: 7.4.4.3.-2 Example 2) request to Object 66: Read /66: TLV payload will contain 2 Object Instances

7.4.5. SenML JSON

When a LwM2M Client is supporting the SenML JSON data format and the format is used to transport Object Instance(s), multiple resource and single resource values for both "Read" and "Write" operations, SenML JSON payload MUST use the format defined in this section. This format MAY be used also 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 JSON format and description.

According to [SENML] semantics, the SenML JSON data format is composed of an array of entries (SenML Records) with optional and mandatory fields.

Each entry of the SenML 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 an entry is a URI path relative to the Base Name; namely the Name attribute could simply be the full URI path of a requested Resource Instance when Base Name is absent.

The SenML JSON format 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 SenML 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 SenML JSON second example below); a given Object Instance MUST appear at most once in that report. The SenML JSON format also includes optional time fields, which allows for multiple versions of representations to be sent in the same payload or indicating the time when the representation was created (e.g., measurement made).

Note: SenML time values (Base Time and Time) are represented in floating point as seconds and a missing time attribute is considered to have a value of zero. Base time and time are added together. In general, positive time values represent an absolute time relative to the Unix epoch, while negative values represent a relative time in the past from the current time. For details see [SENML].

Historical version of notifications are typically generated when "Notification Storing When Disabled or Offline" resource of the LwM2M Server Object is set to true (see Appendix E. LwM2M Objects defined by OMA (Normative)) and when the Device comes on line after having been disabled for a period of time.

The SenML JSON data format has a Media Type application/senml+json.

[LwM2M-TS_1.0] defined the format for application/vnd.oma.lwm2m+json based on an early draft version of SenML. The LwM2M TS 1.1 aligns the SenML JSON format with the final version of the SenML specification.

For backward compatibility, server implementations of LwM2M TS 1.1 MUST support both application/senml+json and application/vnd.oma.lwm2m+json. Client implementations of LwM2M TS 1.1 and later MUST NOT use the application/vnd.oma.lwm2m+json format.

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.
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	vb		Value as a JSON Boolean if the Resource data type is boolean.
Object Link Value	vlo		Value as a JSON string if the Resource data type is Objlnk. Format according to Appendix C. Data Types (Normative) (e.g. "10:03").
Opaque Value	vd		For Resource data type "opaque" this holds the Base64 encoded representation of the Resource.
String Value	vs		Value as a JSON string for all other Resource data types.

Table: 7.4.5.-1 SenML JSON format and description

For example, a request to a Device Object (Read /3/0) of the LwM2M example client would return the following SenML JSON payload. This example has a size of 399 bytes.

[{"bn":"/3/0/","n":"0","vs":"Open Mobile Alliance"},
{"n":"1","vs":"Lightweight M2M Client"},
{"n":"2","vs":"345000123"},
{"n":"3","vs":"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","vs":"+02:00"},
{"n":"16","vs":"U"}]

Table: 7.4.5.-2 SenML JSON payload returned from example request to Device Object (Read /3/0)

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 SenML JSON payload:

[{"bn":"/72/","bt":25462634,"n":"1/2","v":22.4,"t":-5},
{"n":"1/2","v":22.9,"t":-30},
{"n":"1/2","v":24.1,"t":-50}]

Table: 7.4.5.-3 SenML JSON payload from example notification about a Resource containing multiple historical representations of a Temperature Resource

For example, a request to Object 65 of the LwM2M example seen in Figure: C.-1 Object link Resource simple illustration (Read /65/0) would return the following SenML 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 412 bytes.

[{"bn":"/","n":"65/0/0/0","vlo":"66:0"},
{"n":"65/0/0/1","vlo":"66:1"},
{"n":"65/0/1","vs":"8613800755500"},
{"n":"65/0/2","v":1},
{"n":"66/0/0","vs":"myService1"},
{"n":"66/0/1","vs":"Internet.15.234"},
{"n":"66/0/2","vlo":"67:0"},
{"n":"66/1/0","vs":"myService2"},
{"n":"66/1/1","vs":"Internet.15.235"},
{"n":"66/1/2","vlo":"FFFF:FFFF"},
{"n":"67/0/0","vs":"85.76.76.84"},
{"n":"67/0/1","vs":"85.76.255.255"}]

Table: 7.4.5.-4 SenML JSON payload returned from example request to Object 65 of the LwM2M example seen on Figure: C.-1 Object link Resource simple illustration (Read /65/0)

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

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

Table: 7.4.5.-5 SenML JSON payload returned from example request to Device Object on Resource 0 of the LwM2M example client (Read /3/0/0)

For example, a Read-Composite operation from a LwM2M Server to the LwM2M client to read manufacturer name and battery level from the LwM2M Device Object together with the registration lifetime Resource from the LwM2M Server Object will look as follows:

```
[{"n":"/3/0/0"},
{"n":"/3/0/9"},
{"n":"/1/0/1"}]
```

Table: 7.4.5.-6 SenML JSON payload in the server request to read manufacturer name, battery level and registration lifetime

In response to the Read-Composite operation above the LwM2M Client will return a SenML JSON payload similar to the one shown below.

```
[{"n":"/3/0/0", "vs":"Open Mobile Alliance"},
{"n" :"/3/0/9", "v":95},
{"n":"/1/0/1", "v":86400}]
```

Table: 7.4.5.-7 SenML JSON payload in the client response to server request to read manufacturer name, battery level and registration lifetime

For example, a Read-Composite operation from a LwM2M Server to a LwM2M client to read its Location Object together with Network Bearer, Available Network Bearer and Radio Signal Strengt resources in Connectivity Monitoring Object will look as follows:

```
[{"bn":"/4/0/", "n":"0"},
{"n":"1"},
{"n":"2"},
{"bn":"/6/", "n":"0"}]
```

Table: 7.4.5.-8 SenML JSON payload in the server request to read Newtwork Bearer, Available Network Bearer, Radio Signal Strength and Location

Note: As shown above, SenML records used with a Read-Composite operation will not contain any value field while the responses will. This implies the need for SenML parsers to be context aware, i.e. to distinguish between SenML records received in Read-Composite operations and responses to such requests. The media types, application/senml-etch+json and application/senml-etch+cbor, will remove the requirement for context aware parsing.

For example, a Write-Composite operation by a LwM2M Server to switch off 2 light sources, to dim a 3rd to 20% and to set the thermostat to 18 degrees will have a SenML JSON payload as shown in table below. All lights are controlled by

instances of the IPSO Light Control Object (Object ID 3311), while the thermostat is controlled by an instance of the IPSO Set Point Object (Object ID 3308).

```
[{"n":"/3311/0/5850", "vb":false},
{"n":"/3311/1/5850", "vb":false},
{"n":"/3311/2/5851", "v":20},
{"n":"/3308/0/5900", "v":18}]
```

Table: 7.4.5.-9 SenML JSON payload in the server Write-Composite to switch off /3311/0 and /3311/1, while dimming /3311/2

Note: The Write-Composite operation with SenML format can only be used to add or replace a Resource in a Multiple-Instance Resource. The current format does not support the use of a "null" value in a SenML record to indicate the deletion of a specific Resource Instance. The media types, application/senml-etch+json and application/senml-etch+cbor, will remove this constraint.

The following example shows a Read-Composite operation from a LwM2M Server to the LwM2M Client to read all objects. This example offers the same functionality as a Read operation for all objects. Note that a response may not return all objects on the LwM2M Client since access control settings have to be taken into account as well.

```
[{"n":"/"}]
```

Table: 7.4.5.-10 SenML JSON payload in the request to all objects

Next we show an example of a Read-Composite operation issued by a LwM2M Server to the LwM2M Client to read all object instances of the LwM2M Server Object.

```
[{"n":"/1"}]
```

Table: 7.4.5.-11 SenML JSON payload in the request to all object instances of the LwM2M Server Object

The final example illustrates how to request all Resources in the first and the third instance of the LwM2M Server Object using a Read-Composite operation issued by a LwM2M Server to the LwM2M Client.

```
[{"n":"/1/0"},
{"n":"/1/2"}]
```

Table: 7.4.5.-12 SenML JSON payload in the request to all resources of the first and third instance of the LwM2M Server Object

7.4.5.1. LwM2M TS 1.0 JSON format

The JSON format used by the LwM2M TS 1.0 with media type application/vnd.oma.lwm2m+json is formally defined in [LwM2M-TS_1.0]. The JSON payload shown below is an example reply to a request to the Device Object of the LwM2M example Client ("Read /3/0").

{"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"}]
}

7.4.6. SenML CBOR

When a LwM2M Client is supporting the SenML CBOR data format and that format is used to transport Object Instance(s), multiple resource and single resource values for both "Read" and "Write" operations, the SenML CBOR payload MUST use the format defined in Section 6 of [SENML]. The same format MAY be used for transporting a single value of a Resource. The object links use a string map key "vlo" also in SenML CBOR representation.

The example from Table: 7.4.5.-2 SenML JSON payload returned from example request to Device Object (Read /3/0) is shown below in the SenML CBOR format.

90 a3 21 65 2f 33 2f 30 2f 00 61 30 03 74 4f 70
65 6e 20 4d 6f 62 69 6c 65 20 41 6c 6c 69 61 6e
63 65 a2 00 61 31 03 76 4c 69 67 68 74 77 65 69
67 68 74 20 4d 32 4d 20 43 6c 69 65 6e 74 a2 00
61 32 03 69 33 34 35 30 30 30 31 32 33 a2 00 61
33 03 63 31 2e 30 a2 00 63 36 2f 30 02 01 a2 00
63 36 2f 31 02 05 a2 00 63 37 2f 30 02 19 0e d8
a2 00 63 37 2f 31 02 19 13 88 a2 00 63 38 2f 30
02 18 7d a2 00 63 38 2f 31 02 19 03 84 a2 00 61
39 02 18 64 a2 00 62 31 30 02 0f a2 00 64 31 31
2f 30 02 00 a2 00 62 31 33 02 1a 51 82 42 8f a2
00 62 31 34 03 66 2b 30 32 3a 30 30 a2 00 62 31
36 03 61 55

For details see Table: 7.4.5.-2 SenML JSON payload returned from example request to Device Object (Read /3/0).

The same example using the SenML CBOR diagnostic notation is shown below.

[{-2: "/3/0/", 0: "0", 3: "Open Mobile Alliance"},
{0: "1", 3: "Lightweight M2M Client"},
{0: "2", 3: "345000123"},
{0: "3", 3: "1.0"},
{0: "6/0", 2: 1},
{0: "6/1", 2: 5},
{0: "7/0", 2: 3800},
{0: "7/1", 2: 5000},

```
{0: "8/0", 2: 125},
{0: "8/1", 2: 900},
{0: "9", 2: 100},
{0: "10", 2: 15},
{0: "11/0", 2: 0},
{0: "13", 2: 1367491215},
{0: "14", 3: "+02:00"},
{0: "16", 3: "U"}]
```

8. Access Control

When a LwM2M Client interacts with multiple LwM2M Servers there is a need to determine which operation on a certain Object or Object Instance is authorized for which LwM2M Server. The Access Control Object has been designed to offer this access management capability.

A LwM2M Client can be implemented and configured in two ways:

- In the "Access Control-Enabled" configuration the LwM2M Client is capable of granting access rights to one or several LwM2M Servers. When a LwM2M Client is connected to multiple LwM2M Servers in this configuration then the Access Control Object MUST be instantiated unless all LwM2M Servers are granted full access rights on all Objects and Object Instances in the LwM2M Client.
- In the "Access Control-Disabled" configuration the LwM2M Client is only able to interact with a single LwM2M Server. Hence, the capability to manage access rights for multiple LwM2M Servers is not supported, i.e., the Access Control Object is not instantiated. In this configuration the LwM2M Server implicitly has full access rights on all Objects and Object Instances in the LwM2M Client. The Access Control Object need not be instantiated when a LwM2M Client is connected to a single LwM2M Server (i.e., a single LwM2M Server Account exists in the LwM2M Client).

Subsequent sections use the terms "Access Control-Enabled" and "Access Control-Disabled" to refer to these two types of configurations.

8.1. Access Control Object

8.1.1. 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. Provisioning the necessary authorization policies can only be done by the LwM2M Bootstrap-Server.

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 for that Object Instance.

The Access Control Object is specified in Section E.3 LwM2M Object: Access Control and examples are presented in Appendix F. Example LwM2M Client (Informative).

8.1.2. Access Control Object Management

8.1.2.1. Access Control on Object

To authorize a LwM2M Server to instantiate a certain Object supported by the LwM2M Client, not only an Access Control Object Instance - as defined in Section Table: 8.1.2.1.-1 Access Control on Object - MUST be associated to such an Object, but this AC Object Instance MUST also contained an ACL Resource Instance targeting the authorized LwM2M Server.

This kind of Access Control Object Instance associated with a certain Object, MUST only be created or updated during a Bootstrap Phase. The absence of such an association for a certain Object, prevents this Object to be instantiated by any

LwM2M Server.

When such an 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 with the "Create" ("C") access right as ACL Resource value.

Resource ID	Resource Name	Value
0	**Object ID**	ID of the targeted Object
1	**Object Instance ID**	MAX_ID=65535 (meaning: Access Control Object Instance is for the Object Level)
2	**ACL**	A Resource Instance per LwM2M Server authorized to instantiate the Object 5th LSB: "Create" is only configured
3	**Access Control Owner**	MAX_ID=65535 (meaning: managed by Bootstrap Interface)

Table: 8.1.2.1.-1 Access Control on Object

8.1.2.2. Access Control on Object Instance

For being able to register which operations MAY be performed on an Object Instance by a certain LwM2M Server, this Object Instance MUST be associated to an Access Control Object Instance as defined in Section Table: 8.1.2.2.-1 Access Control on 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 8.2. 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 ID	Resource Name	Value
0	**Object ID**	ID of the targeted Object
1	**Object Instance ID**	ID of the newly created Object Instance
2	**ACL**	Any combination of the Access Right {none,R,W,E,D} is acceptable (E.3 LwM2M Object: Access Control)
3	**Access Control Owner**	The Short Server ID of the LwM2M Server owner of the associated Object Instance

Table: 8.1.2.2.-1 Access Control on 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,
    1. 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
    2. 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.

Figure: 8.1.2.2.-1 Illustration of the relations between the LwM2M Access Control Object and the other LwM2M Objects illustrates the Access Control Management of an Object Instance (/X/2) supported by a LwM2M Client. An Access Control Object Instance (/2/Y) is declared in ② which defines the Access Rights applicable to the Instance 2 of the Object X pointed in ①.

In this Access Control Object Instance ②, 4 Instances of the ACL Resource are defined:

- ACL Instance 101 contains the operations granted to the Server having 101 as Short ID (Instance ID:2 of the Server Object ID:1 as pointed in ③)
- ACL Instance 22 contains the operations granted to the Server having 22 as Short ID
- ACL Instance 31 contains the operations granted to the Server having 31 as Short ID
- ACL Instance 0 contains the operations granted by default to any Server for which a specific ACL Instance is not defined in this Instance of the Access Control Object (default Access Rights).

This Access Control Object Instance ② also contains a reference to a Server (Access Control Owner) which is the only one Server authorized to manage that Instance of the Access Control Object.

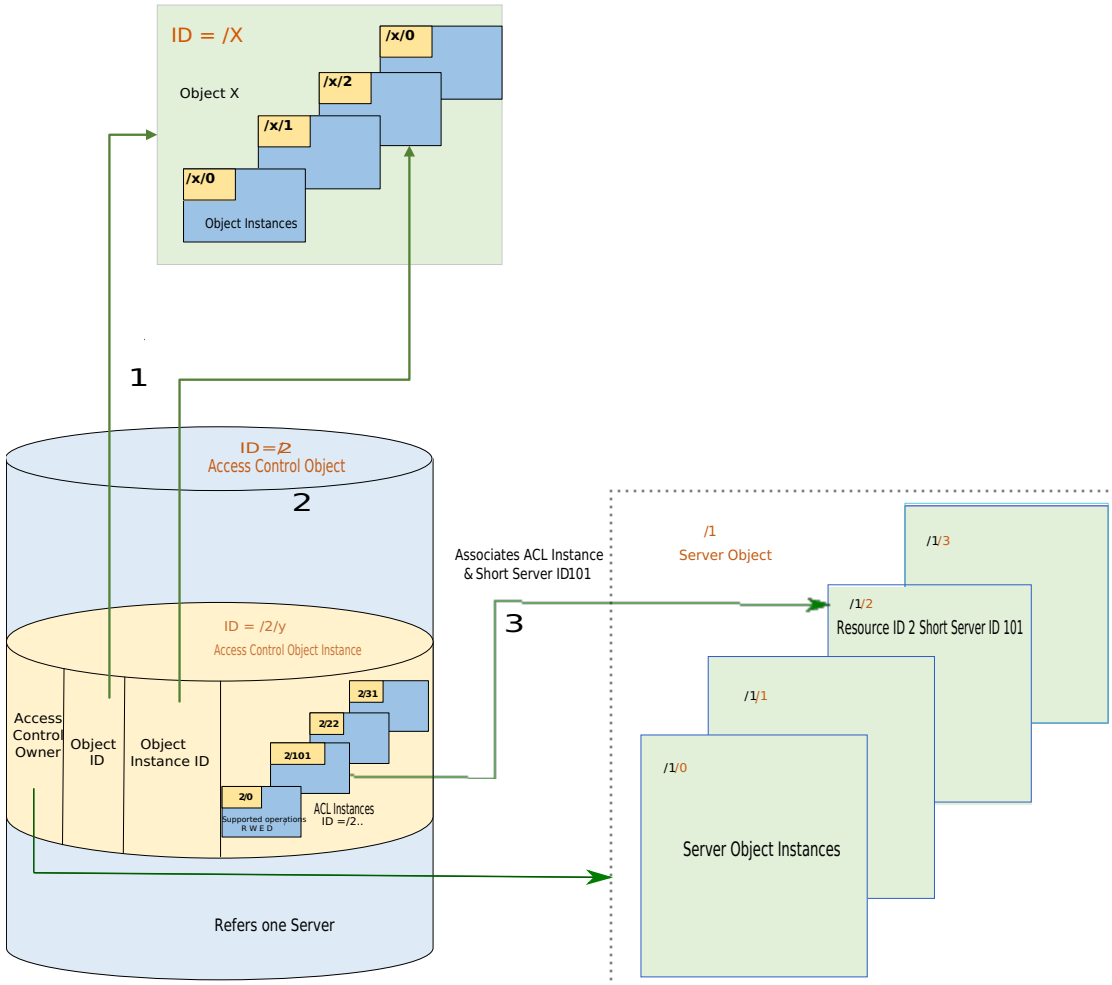


Figure: 8.1.2.2.-1 Illustration of the relations between the LwM2M Access Control Object and the other LwM2M Objects

8.1.3. LwM2M Access Control Context Switch

In an "Access Control-Disabled" context, the Access Control Object is not instantiated. In that case, adding a new LwM2M Server Account, requires the LwM2M Client to switch from the current "Access Control-Disabled" context to an "Access Control-Enabled" context which means the Access Control Object MUST be instantiated. This context switch must be managed in a Bootstrap Phase.

Note: The "Bootstrap-Discover" operation allows to retrieve data regarding the configuration of the initial "Access Control-Disabled" context (list of Objects, Object Instances and Short Server IDs). The Bootstrap-Server has then all information to setup a proper "Access Control-Enabled" context from the initial context one.

8.2. Authorization

For authorizing an operation to proceed on Object Instance(s) or Resource(s), the LwM2M Client MUST:

1. obtain the access rights for the involved Object Instances: * in "Access Control-Disabled" context, full access rights are granted to the LwM2M Server * in "Access Control-Enabled" context, access rights MUST be granted to the LwM2M Server according to Section 8.2.1. Obtaining Access Right

2. check if that access rights granted to the Server are sufficient for the requested operation according to Table: 8.2.-1 Authorization
3. verify –when the check in 2. on access rights is successful– if the requested operation is supported by the targeted Resource(s) (details are provided in Section 8.2.2. Operation on Resource(s) and Object Instance(s) and Section 8.2.3. Operation on Object).

The LwM2M Object specification defines which operations are allowed to be performed on Resource within an Object Instance, see supported operations in Section [LwM2M Object Template and Guidelines]. The operations allowed on a given Resource MUST apply to all the Resource Instances of that Resource.

LwM2M Operations	Minimum Access Right
READ – READ-COMPOSITE - OBSERVE – OBSERVE-COMPOSITE - WRITE-ATTRIBUTE	R
WRITE - WRITE-COMPOSITE	W
DISCOVER	-
DELETE	D
EXECUTE	E

Table: 8.2.-1 Authorization

8.2.1. Obtaining Access Right

In "Access Control-Disabled" context, the LwM2M Server get full access rights.

In "Access Control-Enabled" context:

- for "Create" operation sent by the LwM2M Server, the LwM2M Client MUST get the access right from the ACL Resource Instance corresponding to this LwM2M Server and located in the Access Control Object Instance associated to the targeted Object. Such an Access Control Object Instance – if it exists – has been provisioned during a Bootstrap Phase (Access Control Owner is MAX_ID=65535). If this access right does not have the "Create" value, or cannot be obtained, the LwM2M Server has no access right.
- For operations, except the "Create" operation, the LwM2M Client MUST retrieve the Access Control Object Instance associated with the Object Instance the LwM2M Server has requested access to, and MUST proceed according to the sequence below:
    1. If the LwM2M Server is declared as the Access Control Owner of this Object Instance and there is no ACL Resource Instance for that LwM2M Server, then the LwM2M Client gets full access right.
    2. If the LwM2M Client has an ACL Resource Instance for the LwM2M Server, the LwM2M Client gets the access right from that ACL Resource Instance.
    3. If the LwM2M Server is not declared as the Access Control Owner of this Object Instance and the LwM2M Client does not have the ACL Resource Instance for that Server, the LwM2M Client gets the access right from the ACL Resource Instance (ID:0) containing the default access rights if it exists (Section 8.1.2.2. Access Control on Object Instance).
    4. If the LwM2M Client does not have the ACL Resource Instance ID:0 containing the default access rights, then

the LwM2M Server has no access rights.

5. In case of a Composite operation (Read-Composite, Write-Composite, Observe-Composite), the previous four conditions apply to determine the access right for any Object Instance targeted by the Composite Operation. Composite operations have additional characteristics to determine the final access right granted to the LwM2M Server:

- due to the atomic nature of the Write-Composite operation, all Object Instances involved in that operation MUST have the "Write" access right granted to the requesting LwM2M Server to establish that the "Write" access right is globally granted to such a LwM2M Server for that operation; otherwise, the LwM2M Server has no access right granted.
- due to the non-atomic nature of the Read-Composite and Observe-Composite operations, any Object Instance involved in the requested operation which has no "Read" access right granted to the requesting LwM2M Server MUST be ignored. If at least one of the Object Instances involved in the requested operation has a "Read" access right granted to the LwM2M Server, the "Read" access right is globally granted to such a LwM2M Server for that operation; otherwise the LwM2M Server has no access right granted.

8.2.2. Operation on Resource(s) and Object Instance(s)

When the LwM2M Server targets Resource(s) or Object Instance(s) in a operation, the LwM2M Client MUST obtain an access right for that Server on the Object Instance(s) involved in that operation according to Section 8.2.1. Obtaining Access Right. The LwM2M Client MUST then check if the access right is granted prior to performing the requested operation. If the operation is not permitted, the LwM2M Client MUST send an "Unauthorized" error code to the LwM2M Server.

If the operation is permitted, the following cases apply, according to the requested operation:

- for the "Write" and "Write-Composite" operations, the LwM2M Client MUST perform the operation, only if all the Resources conveyed in the operation are allowed to support 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 operation by sending a "Method not allowed" error code.
- for the "Read" and "Observe" operations, the LwM2M Client MUST retrieve all the Resources except the Resource(s) which does not support the "Read" operation and sends the retrieved Resource(s) information to the LwM2M Server.
- for the "Read-Composite" and "Observe-Composite" operations, and for all Object Instances which have not been ignored during the access right determination (see Section 8.2.1. Obtaining Access Right), the LwM2M Client MUST retrieve all the requested Resources supporting the "Read" operation and sends such retrieved Resource(s) information to the LwM2M Server.
- for the "Execute" operation, if the Resource ID is not specified, the LwM2M Client MUST NOT perform the operation and MUST send an "Operation is not supported" error code to the LwM2M Server. If the Resource ID is specified, the LwM2M Client MUST perform the operation.
- for the "Delete", "Write-Attributes", and "Discover" operations, the LwM2M Client MUST perform the operation.

8.2.3. 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 a "Method Not Allowed" 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 of the Object according to Section 8.2.1. 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 of the Object only if all mandatory Resources are present in the "New Value" parameter (see Section 6. Interfaces). If any of 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 does not support them. The values of the Read-only Resources MUST be setup by the LwM2M Client. If a value of a Read-only Resource is present in the "New Value" parameter then this value MUST be ignored. If the payload (New Value) conveys an Object Instance ID in conflict with one already present in the LwM2M Client then the entire request MUST be rejected and a "Bad Request" error code MUST be returned.

No access rights are needed for the "Discover" operation on an Object, i.e. 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 8.2.1. Obtaining Access Right and the LwM2M Client MUST retrieve all the Object Instances for which the LwM2M Server has the "Read" access right. For each of these qualified Object Instances, the LwM2M Client MUST retrieve all Resources except the Resources which do not support the "Read" operation. The LwM2M Client MUST then aggregate the information produced by the operation on each of these Object Instances individually and then send it to the LwM2M Server.

For the "Write-Attributes" operation, the LwM2M Client MUST perform the operation.

8.2.4. 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.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-LightweightM2M_Core-V1_1-20180710-A	10 Jul 2018	Status changed to Approved by DMSE WG on 10 Jul 2018.
OMA-TS-LightweightM2M_Core-V1_1_1-20190617-A	17 Jun 2019	Status changed to Approved by DMSE WG on 17 Jun 2019.

Table: A.1-1 Approved Version History

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 Attributes

Item	Function	Reference	Requirement
LwM2M-ATTR-001-C-M	Support of Attributes	5.1. Attributes	
LwM2M-ATTR-002-C-M	Support of attachment of Attributes	5.1.1. Attributes Definitions and Rules	
LwM2M-ATTR-003-C—M	Support of characteristics of Attributes in Table 5.1.1. Attributes Definitions and Rules	5.1.1. Attributes Definitions and Rules	
LwM2M-ATTR-004-C-M	Support of <PROPERTIES> Class Attributes	5.1.2. Attributes Classification	
LwM2M-ATTR-005-C-M	Support of <NOTIFICATION> Class Attributes	5.1.2. Attributes Classification	

Table: B.1.1-1 Attributes

B.1.2 Interfaces

Item	Function	Reference	Requirement
LwM2M-INTR-001-C-M	Support of relationships as indicated by Table 6.-1	6. Interfaces	

Table: B.1.2-1 Interfaces

B.1.3 Bootstrap Interface

Item	Function	Reference	Requirement
LwM2M-BOOT-001-C-M	Support of at least one Bootstrap Mode	6. Interfaces	
LwM2M-BOOT-001a-C-O	Support at least pre-provisioning of one LwM2M Bootstrap-Server Account	6. Interfaces	

LwM2M-BOOT-002-C-O	Support of Factory Bootstrap Mode	6.1.3.1. Factory Bootstrap	
LwM2M-BOOT-003-C-O	Support of Bootstrap from Smartcard	6.1.3.2. Bootstrap from Smartcard	LwM2M-BOOT-012C-O
LwM2M-BOOT-004-C-O	Support of Client Initiated Bootstrap	6.1.3.3. Client Initiated Bootstrap	
LwM2M-BOOT-005-C-O	Support of Server Initiated Bootstrap	6.1.3.4. Server Initiated Bootstrap	
LwM2M-BOOT-006-C-M	Support of LwM2M Server Bootstrap Information	6.1.2. Bootstrap Information	
LwM2M-BOOT-007-C-O	Support of LwM2M Bootstrap-Server Bootstrap Information	6.1.2. Bootstrap Information	
LwM2M-BOOT-008-C-M	Support of accepting Bootstrap Information transferred	6.1.2. Bootstrap Information	
LwM2M-BOOT-009-C-M	Support of Bootstrap Sequence	6.1.4. Bootstrap Sequence	
LwM2M-BOOT-010-C-M	Support of Bootstrap Security	6.1.5. Bootstrap Security	
LwM2M-BOOT-011-C-O	Support of Bootstrap from Smartcard with Secure Channel	6. Interfaces, Appendix G. Storage of LwM2M Bootstrap Information on the Smartcard (Normative)	LwM2M-BOOT-012C-O AND LwM2M-SEC-007-C-O
LwM2M-BOOT-012-C-O	Retrieve & Process bootstrap data from Smartcard	6.1.3.2. Bootstrap from Smartcard	
LwM2M-BOOT-013-C-O	Check for Bootstrap Data change in Smartcard	6.1.3.2. Bootstrap from Smartcard	
LwM2M-BOOT-014-C-O	Support of the BOOSTRAP-REQUEST operation	6.1.7.1. Bootstrap-Request Operation	LwM2M-BOOT-004-C-O
LwM2M-BOOT-015-C-O	Support of the BOOTSTRAP-FINISH, BOOTSTRAP DISCOVER, BOOTSTRAP READ, BOOTSTRAP WRITE, BOOTSTRAP DELETE operations	6.1.7.1. Bootstrap-Request Operation 6.1.7.2. Bootstrap-Finish Operation 6.1.7.3. Bootstrap-Discover Operation 6.1.7.4. Bootstrap-Read Operation 6.1.7.5. Bootstrap-Write Operation 6.1.7.6. Bootstrap-Delete Operation	LwM2M-BOOT-004-C-O OR LwM2M-BOOT-005-C-O

LwM2M-BOOT-016-C-O	Check and report Bootstrap Configuration Inconsistency	6.1.6. Bootstrap and Configuration Consistency	LwM2M-BOOT-004-C-O OR LwM2M-BOOT-005-C-O
LwM2M-BOOT-017-C-O	Client shall support rebootstrapping when bootstrap is not finished within defined period	6.1.6. Bootstrap and Configuration Consistency	
LwM2M-BOOT-018-C-O	Client to flush all pending responses to LwM2M server which are not related to the Bootstrap sequence	6.1.1. LwM2M Bootstrap-Server	

Table: B.1.3-1 Bootstrap Interface

B.1.4 Client Registration

Item	Function	Reference	Requirement
LwM2M-CR-001-C-M	Support of “Register” operation	6.2.1. Register Operation	
LwM2M-CR-002-C-M	Support of Endpoint Client Name parameter	6.2.1. Register Operation	
LwM2M-CR-003-C-M	Support of Lifetime parameter	6.2.1. Register Operation	
LwM2M-CR-004-C-M	Support of LwM2M Version parameter	6.2.1. Register Operation	
LwM2M-CR-005-C-M	Support of Binding Mode parameter	6.2.1. Register Operation 6.2.1.1. Bootstrap and LwM2M Server Registration Mechanisms	
LwM2M-CR-006-C-O	Support of SMS Number parameter	6.2.1. Register Operation	
LwM2M-CR-007-C-M	Support of Object and Object Instances parameter	6.2.1. Register Operation	
LwM2M-CR-008-C-M	Support of “Update” operation	6.2.2. Update Operation	
LwM2M-CR-009-C-O	Support of “De-register” operation	6.2.3. De-register Operation	

LwM2M-CR-010-C-O	Support of Updating Bootstrap Information from Smartcard at Register/Update	6.2.2. Update Operation	(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),Security Object Instances, OSCORE Object (ID:21) and instances in the parameters list	6.2.1. Register Operation	
LwM2M-CR-0012-C-M	Support of Object Versioning in the Object and Object Instances parameter list	6.2.1. Register Operation	
LwM2M-CR-013-C-O	Client to support Registration control resources	6.2.1.1. Bootstrap and LwM2M Server Registration Mechanisms	
LwM2M-CR-014-C-M	Client to support the object ver, if the object is new version	6.2.1. Register Operation	

Table: B.1.4-1 Client Registration

B.1.5 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LwM2M-DMSE-001-C-M	Support of “Read” operation	6.3.1. Read Operation	
LwM2M-DMSE-002-C-M	Support of “Discover” operation	6.3.2. Discover Operation	
LwM2M-DMSE-003-C-M	Support of “Write” operation	6.3.3. Write Operation	
LwM2M-DMSE-004-C-M	Support of “Write-Attributes” operation	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-005-C-O	Support of Minimum Period parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-006-C-O	Support of Maximum Period parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-007-C-O	Support of Greater Than parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-008-C-O	Support of Less Than parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-009-C-O	Support of Step parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-010-C-O	Support of Cancel parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-011-C-M	Support of “Execute” operation	6.3.5. Execute Operation	
LwM2M-DMSE-012-C-M	Support of “Create” operation	6.3.6. Create Operation	
LwM2M-DMSE-013-C-M	Support of “Delete” operation	6.3.7. Delete Operation	
LwM2M-DMSE-014-C-O	Support of “Read-Composite”	6.3.8. Read-Composite Operation	
LwM2M-DMSE-015-C-O	Support of “Write-Composite”	6.3.9. Write-Composite Operation	

LwM2M-DMSE-016-C-O	Support of "Send"	6.4.6. Send Operation	

Table: B.1.5-1 Device Management and Service Enablement Interface

B.1.6 Information Reporting

Item	Function	Reference	Requirement
LwM2M-IR-001-C-M	Support of "Observe" operation	6.4.1. Observe Operation	
LwM2M-IR-002-C-M	Support of "Notify" operation	6.4.2. Notify Operation	
LwM2M-IR-003-C-M	Support of "Cancel Observation" operation	6.4.3. Cancel Observation Operation	
LwM2M-IR-004-C-O	Support of "Observe-Composite"	6.4.4. Observe-Composite Operation	
LwM2M-IR-005-C-O	Support of "Cancel Observation-Composite"	6.4.5. Cancel Observation-Composite Operation	

Table: B.1.6-1 Information Reporting

B.1.7 Identifier, Data Types and Serialization Formats

Item	Function	Reference	Requirement
LwM2M-IDT-001-C-O	Support of Plain Text format	7.4.1. Plain Text	
LwM2M-IDT-002-C-O	Support of Opaque format	7.4.2. Opaque	
LwM2M-IDT-003-C-O	Support of TLV format	7.4.4. TLV	
LwM2M-IDT-004-C-O	Support of LwM2M TS 1.0 JSON format	7.4.5.1. LwM2M TS 1.0 JSON format	
LwM2M-IDT-005-C-O	Support of CBOR format	7.4.3. CBOR	
LwM2M-IDT-006-C-M	Support of Data Types in Appendix C	Appendix C. Data Types (Normative)	
LwM2M-IDT-007-C-M	Support of a unique Client Identifier	7.3. Identifiers	
LwM2M-IDT-008-C-O	Support of new media types SenML JSON, SenML CBOR	7.4. Data Formats for Transferring Resource Information	

Table: B.1.7-1 Identifier, Data Types and Serialization Formats

B.1.8 Mechanism

Item	Function	Reference	Requirement
LwM2M-MEC-001-C-O	Support of Queue Mode	6.2.1.2. Behaviour with Current Transport Binding and Modes	
LwM2M-MEC-002-C-M	Support of UDP Binding	6.2.1.2. Behaviour with Current Transport Binding and Modes	
LwM2M-MEC-003-C-O	Support of SMS Binding	6.2.1.2. Behaviour with Current Transport Binding and Modes	

Table: B.1.8-1 Mechanism

B.1.9 Objects

Item	Function	Reference	Requirement
LwM2M-OBJ-001-C-M	Support of LwM2M Security Object	Appendix E.1 LwM2M Object: LwM2M Security	
LwM2M-OBJ-002-C-M	Support of LwM2M Server Object	Appendix E.2 LwM2M Object: LwM2M Server	
LwM2M-OBJ-003-C-O	Support of Access Control Object	Appendix E.3 LwM2M Object: Access Control	
LwM2M-OBJ-004-C-M	Support of Device Object	Appendix E.4 LwM2M Object: Device	
LwM2M-OBJ-005-C-O	Support of Connectivity Monitoring Object	Appendix E.5 LwM2M Object: Connectivity Monitoring	
LwM2M-OBJ-006-C-O	Support of Firmware Update Object	Appendix E.6 LwM2M Object: Firmware Update	
LwM2M-OBJ-007-C-O	Support of Location Object	Appendix E.7 LwM2M Object: Location	
LwM2M-OBJ-008-C-O	Support of Connectivity Statistics Object	Appendix E.8 LwM2M Object: Connectivity Statistics	
LwM2M-OBJ-009-C-O	Support of OSCORE object	Appendix E.9 LwM2M Object: OSCORE	

Table: B.1.9-1 Objects

B.2 SCR for LwM2M Server

B.2.1 Attributes

Item	Function	Reference	Requirement
LwM2M-ATTR-001-S-M	Support of Attributes	5.1. Attributes	
LwM2M-ATTR-002-S-M	Support of attachment of Attributes	5.1.1. Attributes Definitions and Rules	
LwM2M-ATTR-003-S—M	Support of characteristics of Attributes in Table 5.1.1. Attributes Definitions and Rules.-1	5.1.1. Attributes Definitions and Rules	
LwM2M-ATTR-004-S-M	Support of <PROPERTIES> Class Attributes	5.1.2. Attributes Classification	
LwM2M-ATTR-005-S-M	Support of <NOTIFICATION> Class Attributes	5.1.2. Attributes Classification	

Table: B.2.1-1 Attributes

B.2.2 Interfaces

Item	Function	Reference	Requirement
LwM2M-INTR-001-S-M	Support of relationships as indicated by Table 6.-1	6. Interfaces	

Table: B.2.2-1 Interfaces

B.2.3 Bootstrap Interface

Item	Function	Reference	Requirement
LwM2M-BOOT-005-S-M	Support of Server Initiated Bootstrap	6.1.3.4. Server Initiated Bootstrap	
LwM2M-BOOT-010-S-M	Support of Bootstrap Security	6.1.5. Bootstrap Security	

Table: B.2.3-1 Bootstrap Interface

B.2.4 Client Registration

Item	Function	Reference	Requirement
LwM2M-CR-001-S-M	Support of "Register" operation	6.2.1. Register Operation	
LwM2M-CR-002-S-M	Support of Endpoint Client Name parameter	6.2.1. Register Operation	
LwM2M-CR-003-S-M	Support of Lifetime parameter	6.2.1. Register Operation	
LwM2M-CR-004-S-M	Support of LwM2M Version parameter	6.2.1. Register Operation	
LwM2M-CR-005-S-M	Support of Binding Mode parameter	6.2.1. Register Operation 6.2.1.1. Bootstrap and LwM2M Server Registration Mechanisms	
LwM2M-CR-006-S-M	Support of SMS Number parameter	6.2.1. Register Operation	
LwM2M-CR-007-S-M	Support of Object and Object Instances parameter	6.2.1. Register Operation	

LwM2M-CR-008-S-M	Support of “Update” operation	6.2.2. Update Operation	
LwM2M-CR-009-S-M	Support of “De-register” operation	6.2.3. De-register Operation	
LwM2M-CR-010-S-M	Support removal of the Client and existing observations, when no update is received	6. Interfaces	
LwM2M-CR-011-S-O	Server to support Registration control resources	6.2.1.1. Bootstrap and LwM2M Server Registration Mechanisms	

Table: B.2.4-1 Client Registration

B.2.5 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LwM2M-DMSE-001-S-M	Support of “Read” operation	6.3.1. Read Operation	
LwM2M-DMSE-002-S-M	Support of “Discover” operation	6.3.2. Discover Operation	
LwM2M-DMSE-003-S-M	Support of “Write” operation	6.3.3. Write Operation	
LwM2M-DMSE-004-S-M	Support of “Write-Attributes” operation	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-005-S-M	Support of Minimum Period parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-006-S-M	Support of Maximum Period parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-007-S-M	Support of Greater Than parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-008-S-M	Support of Less Than parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-009-S-M	Support of Step parameter	6.3.4. Write-Attributes Operation	
LwM2M-DMSE-010-S-M	Support of “Execute” operation	6.3.5. Execute Operation	
LwM2M-DMSE-011-S-M	Support of “Create” operation	6.3.6. Create Operation	
LwM2M-DMSE-012-S-M	Support of “Delete” operation	6.3.7. Delete Operation	
LwM2M-DMSE-013-S-O	Support of “Read-Composite”	6.3.8. Read-Composite Operation	
LwM2M-DMSE-014-S-O	Support of “Write-Composite”	6.3.9. Write-Composite Operation	
LwM2M-DMSE-015-S-O	Support of “Send”	6.4.6. Send Operation	

Table: B.2.5-1 Device Management and Service Enablement Interface

B.2.6 Information Reporting

Item	Function	Reference	Requirement
LwM2M-IR-001-S-M	Support of “Observe” operation	6.4.1. Observe Operation	

LwM2M-IR-002-S-M	Support of “Notify” operation	6.4.2. Notify Operation	
LwM2M-IR-003-S-M	Support of “Cancel Observation” operation	6.4.3. Cancel Observation Operation	
LwM2M-IR-004-S-O	Support of “Observe-Composite”	6.4.4. Observe-Composite Operation	
LwM2M-IR-005-S-O	Support of “Cancel Observation-Composite”	6.4.5. Cancel Observation-Composite Operation	

Table: B.2.6-1 Information Reporting

B.2.7 Identifier, Data Types and Serialization Formats

Item	Function	Reference	Requirement
LwM2M-IDT-001-S-M	Support of Plain Text format	7.4.1. Plain Text	
LwM2M-IDT-002-S-M	Support of Opaque format	7.4.2. Opaque	
LwM2M-IDT-003-S-M	Support of TLV format	7.4.4. TLV	
LwM2M-IDT-004-S-M	Support of JSON format	7.4.5.1. LwM2M TS 1.0 JSON format	
LwM2M-IDT-005-S-M	Support of CBOR format	7.4.3. CBOR	
LwM2M-IDT-006-S-M	Support of Data Types in Appendix C	Appendix C. Data Types (Normative)	
LwM2M-IDT-007-S-M	Support of a unique Client Identifier	7.3. Identifiers	
LwM2M-IDT-008-S-M	Support of new media types SenML JSON, SenML CBOR	7.4. Data Formats for Transferring Resource Information	

Table: B.2.7-1 Identifier, Data Types and Serialization Formats

B.2.8 Mechanism

Item	Function	Reference	Requirement
LwM2M-MEC-001-S-M	Support of Queue Mode	6.2.1.2. Behaviour with Current Transport Binding and Modes	
LwM2M-MEC-002-S-M	Support of UDP Binding	6.2.1.2. Behaviour with Current Transport Binding and Modes	
LwM2M-MEC-003-S-O	Support of SMS Binding	6.2.1.2. Behaviour with Current Transport Binding and Modes	

Table: B.2.8-1 Mechanism

B.2.9 Objects

Item	Function	Reference	Requirement
LwM2M-OBJ-001-S-M	Support of LwM2M Security Object	Appendix E.1 LwM2M Object: LwM2M Security	
LwM2M-OBJ-002-S-M	Support of LwM2M Server Object	Appendix E.2 LwM2M Object: LwM2M Server	
LwM2M-OBJ-003-S-O	Support of Access Control Object	Appendix E.3 LwM2M Object: Access Control	
LwM2M-OBJ-004-S-M	Support of Device Object	Appendix E.4 LwM2M Object: Device	
LwM2M-OBJ-005-S-O	Support of Connectivity Monitoring Object	Appendix E.5 LwM2M Object: Connectivity Monitoring	
LwM2M-OBJ-006-S-O	Support of Firmware Update Object	Appendix E.6 LwM2M Object: Firmware Update	
LwM2M-OBJ-007-S-O	Support of Location Object	Appendix E.7 LwM2M Object: Location	
LwM2M-OBJ-008-S-O	Support of Connectivity Statistics Object	Appendix E.8 LwM2M Object: Connectivity Statistics	
LwM2M-OBJ-009-S-O	Support of OSCORE object	Appendix E.9 LwM2M Object: OSCORE	

Table: B.2.9-1 Objects

B.3 SCR for LwM2M Bootstrap Server

B.3.1 Bootstrap Interface

Item	Function	Reference	Requirement
LwM2M-BOOT-001-BS-M	Support “Client Initiated Bootstrap”	6. Interfaces	

Table: B.3.1-1 Bootstrap Interface

Appendix C. Data Types (Normative)

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

Data Type	Description
String	A UTF-8 string, the minimum and/or maximum length of the String MAY be defined.
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.
Unsigned Integer	An 8, 16, 32 or 64-bit unsigned integer. The valid range of the value for a Resource SHOULD be defined. This data type may also used as a bitmask whereby bit positions range from 0 to sizeof(unsigned integer)-1. For example, an 8 bit unsigned integer can hold a bitmask of 8 "features" ranging from bit(0) to bit(7) whereby each bit position, when set, corresponds to the unsigned integer value 2^(bit position). The following example shows an 8-bit unsigned integer value illustrates a bitmask with "0001 0010" whereby bit(1) and bit(4) are set resulting in the unsigned integer value 18 (i.e., 2^1+2^4).
Float	A 32 or 64-bit floating point value. The valid range of the value for a Resource SHOULD be defined.
Boolean	An 8 bit unsigned integer with the value 0 for False and the value 1 for True.
Opaque	A sequence of binary octets, the minimum and/or maximum length of the String MAY be defined.
Time	Unix Time. A signed integer representing the number of seconds since Jan 1st, 1970 in the UTC time zone.
Objlnk	Object Link. The object link is used to refer to an Instance of a given Object. An Object Link referencing no Object Instance will contain 2 MAX-ID values (null link).
Corelnk	CoRE Link. A link is used to refer to Resources on a LWM2M Client and their attributes as specified in [RFC6690].
none	No specific data type affected to that resource: it exclusively concerns Executable Resource.

Table: C.-1 Data Types

An Object Link referencing no Object Instance will contain 2 MAX-ID values (null link).

Data Type	Text Format	TLV Format	JSON SenML Format	CBOR SenML Format
String	Represented as a UTF-8 string.	Represented as a UTF-8 string of Length bytes.	Represented as a string.	Represented as a UTF-8 text string, as defined in Section 2 of [RFC7049].
Integer	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).	Represented as a number.	Represented as an integer, as defined in Section 2 of [RFC7049].

Unsigned Integer	Represented as an ASCII unsigned integer. For example, the unsigned integer value 18 results in a 2 character/byte long ASCII string "18".	Represented as a binary unsigned integer in network byte order. 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).	Represented as a number.	Represented as an unsigned integer, as defined in Section 2 of [RFC7049].
Float	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).	Represented as a number.	Represented as a floating-point number, as defined in Section 2 and Section 2.3 of [RFC7049].
Boolean	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.	Represented as the JSON boolean data type, which is either represented as the string 'true' or 'false'.	Represented as the value 'True' or 'False', as defined in Table 2 of [RFC7049].
Opaque	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.	Represented as a Base64 encoded representation of the data in a JSON string.	Represented as a byte string, as defined in Section 2 of [RFC7049].
Time	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.	Represented as a number.	Represented as date/time strings, as defined in Section 2.4.1 of [RFC7049].
Objlnk	Represented as a UTF-8 string containing 2 16-bit ASCII integers separated by a ':' ASCII character. The first one represents the Object ID, and the second one represents the Object Instance ID.	Represented as two concatenated 16 bit unsigned integers following the Network Byte Order convention. The first one represents the Object ID, and the second one represents the Object Instance ID. This value is always 4 bytes long.	Represented as a string containing two 16-bit ASCII integers separated by a ':' ASCII character. The first one represents the Object ID, and the second one represents the Object Instance ID.	Represented as a string containing two 16-bit ASCII integers separated by a ':' ASCII character. The first one represents the Object ID, and the second one represents the Object Instance ID.

Corelnk	Represented as a String using CoRE Link format. For example, an IPSO temperature sensor with measurements query for a resource, with a resource attribute greater than 23: ;gt=23 If the Corelnk refers to an Instance of a given Object (as an Objlnk), then the same validity check as with Objlnk is required.	Same representation as String.	Represented as a string using the CoRE Link format.	Represented as a UTF-8 text string, as defined in Section 2 of [RFC7049].
none	Not applicable	Not applicable	The value is either represented as a string 'null' or as an empty field.	Represented as the value 'Null', as defined in Table 2 of [RFC7049].

Table: C.-2 Data Type Mapping

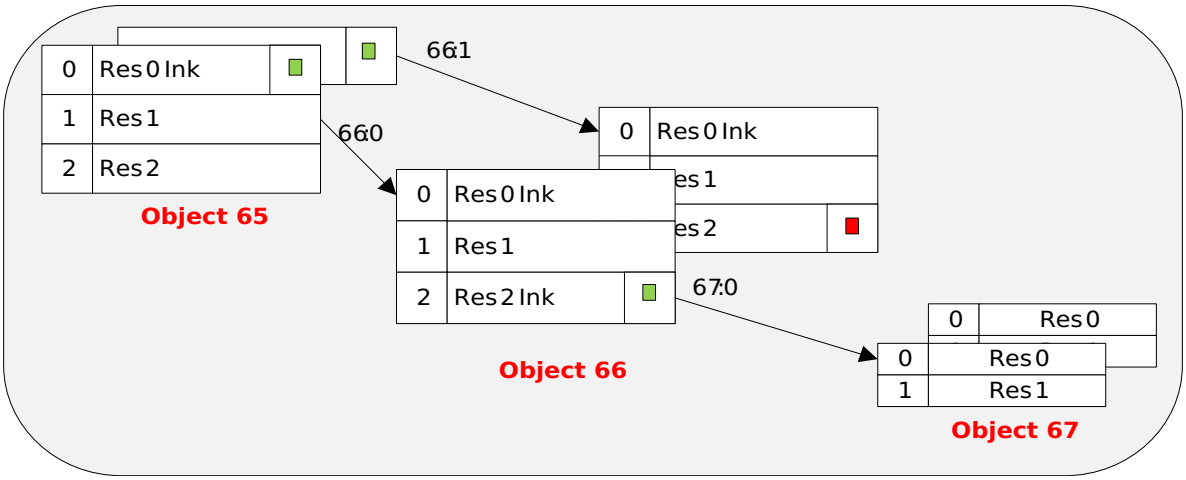


Figure: C.-1 Object link Resource simple illustration

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 LwM2M Object: *<LwM2M object name>*

Description

Object definition:

Name	Object ID	Object Version	LWM2M Version
Object Name	16-bit Unsigned Integer	Object version, e.g.: 1.0	LwM2m version, e.g.: 1.1
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:{oma,ext,x}:{Object ID}[:{version}]		Multiple/Single	Mandatory/Optional

Table: D.1-1 Object definition

- **Name**: specifies the Object name.
- **Object ID**: specifies the Object ID.
- **Object Version**: specifies the version of the Object. The first version of an Object must be version "1.0".
- **LWM2M Version**: specifies the version of the LwM2M protocol, e.g. version "1.1".
- **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 Instances 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, the Object URN of the 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, Unsigned Integer, Float, Boolean, Opaque, Time, Objlnk Corelnk none	If any	If any	Description

Table: D.1-2 Resource definition

- **ID**: specifies the Resource ID, which is unique within an Object or globally unique in case of Reusable Resources.
- **Name**: specifies the Resource name.
- **Operations**: indicates which operations the Resource supports in the given interface. This field can be set to a combination of Read (R), Write (W), and Execute (E). The Execute Operation cannot be used together with the Read and Write operations. This field may also have an empty value, which means that this field can only be accesses by the "Bootstrap" interface. More information about the interfaces and their operations can be found in Section 6. Interfaces.
- **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. A Resource, which supports the "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. A Resource, which supports the "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 possible values of a Resource. A range is defined using two numeric values separated by two dots ("..") to indicate the lower and upper limits (inclusive). Enumeration is defined using one or more comma separated values. For the opaque data type, the value in this field represents constraints on the length of the Resource expressed in octets. For example, `1..42` indicates that the Resource can be 1 to 42 octets long. A value of `5,10,15` in this field indicates that the Resource can be 5, 10, or 15 octets long. A value of `8` in this field indicates that the Resource is always 8 octets long. For the string data type, a numeric range or enumeration has the same meaning as the opaque data type. Additionally, a comma separated list of quoted strings indicates the enumerated string values. For example, `"foo","bar"` indicates that the string Resource can only have the value `foo` or `bar`.
- **Units**: specifies the unit of the Resource value.
- **Description**: specifies the Resource description.

In addition to the object and resource templates described in the previous two tables of this section it is also important to document the syntax and semantic of the arguments of Executable Resources. The table below illustrates the method for documenting the arguments based on the plain text format following the ABNF format described in Section Execute.

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

Table: D.1-3 Executable Resource Arguments Definition

An example can be found with the Portfolio Object.

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. The rules for Object registry are documented at: http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

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 rules for registering Resource are documented at:
http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

Appendix E. LwM2M Objects defined by OMA (Normative)

This appendix provides LwM2M Objects defined by OMA SpecWorks that are a core part of the LwM2M specification. Other organizations and companies may define additional LwM2M Objects according to the guidelines provided in Appendix D. LwM2M Object Template and Guidelines (Normative).

The following LwM2M Object descriptions are contained in this appendix of the LwM2M v1.1 specification. Other objects can also be used with this version of LwM2M v1.1 and can be found in the OMNA registry.

Object Version	Object ID
LwM2M Security v1.1	0
LwM2M Server v1.1	1
LwM2M Access Control v1.0	2
LwM2M Device v1.1	3
LwM2M Connectivity Monitoring v1.2	4
LwM2M Firmware Update v1.0	5
LwM2M Location v1.0	6
LwM2M Connectivity Statistics v1.0	7
LwM2M OSCORE v1.0	21

Table: E.-1 LwM2M Objects

The LwM2M Server MUST support the Security, Server, and Device Objects. The LwM2M Server SHOULD support the Access Control, Connectivity, Firmware Update, Location, and Connectivity Statistics Objects.

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	Object Version	LWM2M Version
LWM2M Security	0	1.1	1.1
Object URN		Instances	Mandatory

urn:oma:lwm2m:oma:0:1.1	Multiple	Mandatory

Table: E.1-1 LwM2M Object: LWM2M Security object definition

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 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, public key (RPK mode) or PSK Identity (PSK mode).
4	Server Public Key		Single	Mandatory	Opaque			Stores the LwM2M Server's, respectively LwM2M Bootstrap-Server's, certificate, public key (RPK mode) or trust anchor. The Certificate Mode Resource determines the content of this resource.
5	Secret Key		Single	Mandatory	Opaque			Stores the secret key (PSK mode) or private key (RPK or certificate mode).
6	SMS Security Mode		Single	Optional	Integer	0..255		Determines which SMS security mode is used: 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		Stores the KIc, KID, SPI and TAR.
8	SMS Binding Secret Key(s)		Single	Optional	Opaque	16,32,48		Stores the values of the key(s) for the SMS binding.
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.

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 a value of 'false'. The values ID:0 and ID:65535 values MUST NOT be used for identifying the LwM2M Server.
11	Client Hold Off Time		Single	Optional	Integer		s	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. This information is relevant for use with a Bootstrap-Server only.
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.
13	Matching Type		Single	Optional	Unsigned Integer	0..3	s	The Matching Type Resource specifies how the certificate or raw public key in in the Server Public Key is presented. Four values are currently defined: 0: Exact match. This is the default value and also corresponds to the functionality of LwM2M v1.0. Hence, if this resource is not present then the content of the Server Public Key Resource corresponds to this value. 1: SHA-256 hash [RFC6234] 2: SHA-384 hash [RFC6234] 3: SHA-512 hash [RFC6234]
14	SNI		Single	Optional	String			This resource holds the value of the Server Name Indication (SNI) value to be used during the TLS handshake. When this resource is present then the LwM2M Server URI acts as the address of the service while the SNI value is used for matching a presented certificate, or PSK identity.
15	Certificate Usage		Single	Optional	Unsigned Integer	0..3	s	The Certificate Usage Resource specifies the semantic of the certificate or raw public key stored in the Server Public Key Resource, which is used to match the certificate presented in the TLS/DTLS handshake. The currently defined values are 0 for "CA constraint", 1 for "service certificate constraint", 2 for "trust anchor assertion", and 3 for "domain-issued certificate". When this resource is absent, value (3) for domain issued certificate mode is assumed. More details about the semantic of each value can be found in the security consideration section of the LwM2M specification.

16	DTLS/TLS Ciphersuite		Multiple	Optional	Unsigned Integer			When this resource is present it instructs the TLS/DTLS client to propose the indicated ciphersuite(s) in the ClientHello of the handshake. A ciphersuite is indicated as a 32-bit integer value. The IANA TLS ciphersuite registry is maintained at https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml. As an example, the TLS_PSK_WITH_AES_128_CCM_8 ciphersuite is represented with the following string "0xC0,0xA8". To form an integer value the two values are concatenated. In this example, the value is 0xc0a8 or 49320.
17	OSCORE Security Mode		Single	Optional	Objlnk			If this resource is defined, it provides a link to the OSCORE Object Instance.

Table: E.1-2 LwM2M Object: LWM2M Security Resource definitions

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	Object Version	LWM2M Version
LwM2M Server	1	1.1	1.1
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:oma:1:1.1		Multiple	Mandatory

Table: E.2-1 LwM2M Object: LwM2M Server object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Short Server ID	R	Single	Mandatory	Integer	1..65534		Used as link to associate server Object Instance.
1	Lifetime	RW	Single	Mandatory	Integer		s	Specify the lifetime of the registration in seconds (see Client Registration Interface).

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 are those listed in the LwM2M Core Specification		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. The LwM2M Client can use a transport binding supported in the Current Binding Mode, Preferred Transport resource or the transport specified as an argument in the Registration Update Trigger.

9	Bootstrap-Request Trigger	E	Single	Optional				When this Resource is executed the LwM2M Client MUST initiate a "Client Initiated Bootstrap" procedure in using the LwM2M Bootstrap-Server Account.
10	APN Link	RW	Single	Optional	Objlnk			If this resource is defined, it provides a link to the APN connection profile Object Instance (OMNA registered Object ID:11) to be used to communicate with this server.
11	TLS-DTLS Alert Code	R	Single	Optional	Unsigned Integer	0..255		If this resource is defined, it contains the most recent TLS / DTLS alert message received from the LwM2M Server respective represented by the AlertDescription defined in Section 7.2 of RFC 5246. This resource set by the LwM2M Client may help the LwM2M Bootstrap-Server to determine the cause of TLS/DTLS connection failure with the respective LwM2M Server.
12	Last Bootstrapped	R	Single	Optional	Time			If this resource is defined, it represents the last time that the bootstrap server updated this LwM2M Server Account. The LwM2M Client is responsible for updating this value. When the Bootstrap Server detects that this LwM2M Server Account is "out-of-date", the Bootstrap Server can update the LwM2M Server Account as represented by the LwM2M Server object instance.
13	Registration Priority Order		Single	Optional	Unsigned Integer			The LwM2M Client sequences the LwM2M Server registrations in increasing order of this value. If this value is not defined, registration attempts to this server are not impacted by other server registrations.
14	Initial Registration Delay Timer		Single	Optional	Unsigned Integer		s	The delay before registration is attempted for this LwM2M Server based upon the completion of registration of the previous LwM2M Server in the registration order. This is only applied until the first successful registration after a successful bootstrapping sequence.
15	Registration Failure Block		Single	Optional	Boolean			When set to true and registration to this LwM2M server fails, the LwM2M Client blocks registration to other servers in the order. When set to false, the LwM2M Client proceeds with registration to the next server in the order.
16	Bootstrap on Registration Failure		Single	Optional	Boolean			If set to true, this indicates that the LwM2M Client should re-bootstrap when either registration is explicitly rejected by the LwM2M Server or registration is considered as failing as dictated by the other resource settings. If set to false, the LwM2M Client will continue with the registration attempts as dictated by the other resource settings.

17	Communication Retry Count		Single	Optional	Unsigned Integer			The number of successive communication attempts before which a communication sequence is considered as failed.
18	Communication Retry Timer		Single	Optional	Unsigned Integer		s	The delay between successive communication attempts in a communication sequence. This value is multiplied by two to the power of the communication retry attempt minus one (2** (retry attempt-1)) to create an exponential back-off.
19	Communication Sequence Delay Timer		Single	Optional	Unsigned Integer			The delay between successive communication sequences. A communication sequence is defined as the exhaustion of the Communication Retry Count and Communication Retry Timer values. A communication sequence can be applied to server registrations or bootstrapping attempts. MAX_VALUE means do not perform another communication sequence.
20	Communication Sequence Retry Count		Single	Optional	Unsigned Integer			The number of successive communication sequences before which a registration attempt is considered as failed.
21	Trigger	RW	Single	Optional	Boolean			Using the Trigger Resource a LwM2M Client can indicate whether it is reachable over SMS (value set to 'true') or not (value set to 'false'). The default value (resource not present) is 'false'. When set to 'true' the LwM2M Server MAY, for example, request the LwM2M Client to perform operations, such as the "Update" operation by sending an "Execute" operation on "Registration Update Trigger" Resource via SMS. No SMS response is expected for such a message.
22	Preferred Transport	RW	Single	Optional	String	The possible values are those listed in the LwM2M Core Specification		Only a single transport binding SHALL be present. When the LwM2M client supports multiple transports, it MAY use this transport to initiate a connection. This resource can also be used to switch between multiple transports e.g. a non-IP device can switch to UDP transport to perform firmware updates.
23	Mute Send	RW	Single	Optional	Boolean			If true or the Resource is not present, the LwM2M Client Send command capability is de-activated. If false, the LwM2M Client Send Command capability is activated.

Table: E.2-2 LwM2M Object: LwM2M Server Resource definitions

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	Object Version	LWM2M Version
LwM2M Access Control	2	1.0	1.0
Object URN		Instances	Mandatory
urn:oma:lwm2m:oma:2		Multiple	Optional

Table: E.3-1 LwM2M Object: LwM2M Access Control object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Object ID	R	Single	Mandatory	Integer	1..65534		Resources 0 and 1 point to the Object Instance for which the Instances of the ACL Resource of that Access Control Object Instance are applicable.
1	Object Instance ID	R	Single	Mandatory	Integer	0..65535		See above
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, 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.

Table: E.3-2 LwM2M Object: LwM2M Access Control Resource definitions

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	Object Version	LWM2M Version
Device	3	1.1	1.1
Object URN		Instances	Mandatory
urn:oma:lwm2m:oma:3:1.1		Single	Mandatory

Table: E.4-1 LwM2M Object: Device object definition

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 3: Fuel Cell 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			Present voltage for each Available Power Sources Resource Instance. The unit used for this resource is in mV.
8	Power Source Current	R	Multiple	Optional	Integer			Present current for each Available Power Source. The unit used for this resource is in mA.

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			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. When an error associated with a Resource Instance is no longer present, that Resource Instance is deleted. When the single existing error is no longer present, the LwM2M Client returns to the original no error state where Instance 0 has value 0. 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, then re-evaluate all error conditions and update and create Resources Instances to capture all current error conditions.
13	Current Time	RW	Single	Optional	Time			Current UNIX time of the LwM2M Client. The LwM2M Client should be responsible to increase this time value as every second elapses. The LwM2M Server is able to write this Resource to make the LwM2M Client synchronized with the LwM2M Server.
14	UTC Offset	RW	Single	Optional	String			Indicates the UTC offset currently in effect for this LwM2M Device. UTC+X [ISO 8601].
15	Timezone	RW	Single	Optional	String			Indicates in which time zone the LwM2M Device is located, in IANA Timezone (TZ) database format.
16	Supported Binding and Modes	R	Single	Mandatory	String			Indicates which bindings and modes are supported in the LwM2M Client. The possible values are those listed in the LwM2M Core Specification.
17	Device Type	R	Single	Optional	String			Type of the device (manufacturer specified string: e.g. smart meters / dev Class / …)
18	Hardware Version	R	Single	Optional	String			Current hardware version of the device

19	Software Version	R	Single	Optional	String			Current software version of the device (manufacturer specified string). On elaborated LwM2M device, SW could be split in 2 parts: a firmware one and a higher level software on top. Both pieces of Software are together managed by LwM2M Firmware Update Object (Object ID 5)
20	Battery Status	R	Single	Optional	Integer	0..6		This value is only valid for the Device Internal Battery if present (one Available Power Sources Resource Instance value is 1). Battery Status Meaning Description 0 Normal The battery is operating normally and not on power. 1 Charging The battery is currently charging. 2 Charge Complete The battery is fully charged and still on power. 3 Damaged The battery has some problem. 4 Low Battery The battery is low on charge. 5 Not Installed The battery is not installed. 6 Unknown The battery information is not available.
21	Memory Total	R	Single	Optional	Integer			Total amount of storage space which can store data and software in the LwM2M Device (expressed in kilobytes).
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.

Table: E.4-2 LwM2M Object: Device Resource definitions

Battery Status	Meaning	Description
0	Normal	The battery is operating normally and not on power.
1	Charging	The battery is currently charging.
2	Charge Complete	The battery is fully charged and still on power.
3	Damaged	The battery has some problem.
4	Low Battery	The battery is low on charge.
5	Not Installed	The battery is not installed.
6	Unknown	The battery information is not available.

Table: E.4-3 Battery Status

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	Object Version	LWM2M Version
Connectivity Monitoring	4	1.2	1.1
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:oma:4:1.2		Single	Optional

Table: E.5-1 LwM2M Object: Connectivity Monitoring object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Network Bearer	R	Single	Mandatory	Integer	0..50		Indicates the network bearer used for the current LwM2M communication session from the network bearer list below. The number range is split into three categories: 0 – 20 are Cellular Bearers 21 – 40 are Wireless Bearers 41 – 50 are Wireline Bearers More specifically: 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: NB-IoT 8 – 20: Reserved for other types of cellular network 21: WLAN network 22: Bluetooth network 23: IEEE 802.15.4 network 24 – 40: Reserved for other types of local wireless network 41: Ethernet 42: DSL 43: PLC 44 – 50: reserved for other types of wireline networks.
1	Available Network Bearer	R	Multiple	Mandatory	Integer	0..50		Indicates a 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			Indicates the average value of the received signal strength indication used in the current network bearer (as indicated by Resource 0 of this Object). The value is expressed in dBm. For the following network bearers the signal strength parameters indicated below are represented by this resource: GSM: RSSI UMTS: RSCP LTE: RSRP NB-IoT: NRSRP For more details on Network Measurement Report, refer to the appropriate Cellular or Wireless Network standards, (e.g. for LTE Cellular Network refer to 3GPP 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 0…7, refer to [3GPP 44.018] for more details on Network Measurement Report encoding), RSRQ for LTE, (refer to [3GPP 36.214]), NRSRQ for NB-IoT (refer to [3GPP 36.214]).
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 percentage indicating the average utilization of the link to the next-hop IP router.
7	APN	R	Multiple	Optional	String			Access Point Name in case Network Bearer Resource is a Cellular Network.
8	Cell ID	R	Single	Optional	Integer			Serving Cell ID in case Network Bearer Resource is a Cellular Network. As specified in TS [3GPP 23.003] and in [3GPP. 24.008]. Range (0…65535) in GSM/EDGE UTRAN Cell ID has a length of 28 bits. Cell Identity in WCDMA/TD-SCDMA. Range: (0…268435455). LTE Cell ID has a length of 28 bits. Parameter definitions in [3GPP 25.331].
9	SMNC	R	Single	Optional	Integer	0..999		Serving Mobile Network Code. This is applicable when the Network Bearer Resource value is referring to a cellular network. As specified in TS [3GPP 23.003].
10	SMCC	R	Single	Optional	Integer	0..999		Serving Mobile Country Code. This is applicable when the Network Bearer Resource value is referring to a cellular network. As specified in TS [3GPP 23.003].
11	SignalSNR	R	Single	Optional	Integer		dB	SINR: Signal to Interference plus Noise Ratio SINR is the ratio of the strength of the received signal to the strength of the received interference signal (noise and interference).
12	LAC	R	Single	Optional	Integer			Location Area Code in case Network Bearer Resource is a Cellular Network. As specified in TS [3GPP 23.003] and in [3GPP. 24.008]

Table: E.5-2 LwM2M Object: Connectivity Monitoring Resource definitions

E.6 LwM2M Object: Firmware Update

Description

This LwM2M Object enables management of firmware which is to be updated. This Object includes installing a 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 is to allow a LwM2M Client to connect to any LwM2M Server to obtain a firmware image using the object and resource structure defined in this section experiencing communication security

protection using TLS/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 images 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 this version of the specification. A LwM2M Server may also instruct a LwM2M Client to fetch a firmware image from a dedicated server (instead of pushing firmware images 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	Object Version	LWM2M Version
Firmware Update	5	1.0	1.0
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:oma:5		Single	Optional

Table: E.6-1 LwM2M Object: Firmware Update object definition

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		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				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 has completed, or, if the device has downloaded the firmware package from the Package URI the state changes to Downloaded. Writing an empty string to Package URI Resource or setting the Package Resource to NULL ('\0'), 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. The firmware update state machine is illustrated in Figure 29 of the LwM2M version 1.0 specification (and also in Figure E.6.1-1 of this 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 image 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		Name of the Firmware Package
7	PkgVersion	R	Single	Optional	String	0..255		Version of the Firmware package

8	Firmware Update Protocol Support	R	Multiple	Optional	Integer	0..5		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) 4: CoAP over TCP (as defined in RFC 8323) 5: CoAP over TLS (as defined in RFC 8323) 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	0..2		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.

Table: E.6-2 LwM2M Object: Firmware Update Resource definitions

E.6.1 Firmware Update State Machine

Figure: E.6.1-1 Firmware Update Mechanisms 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.

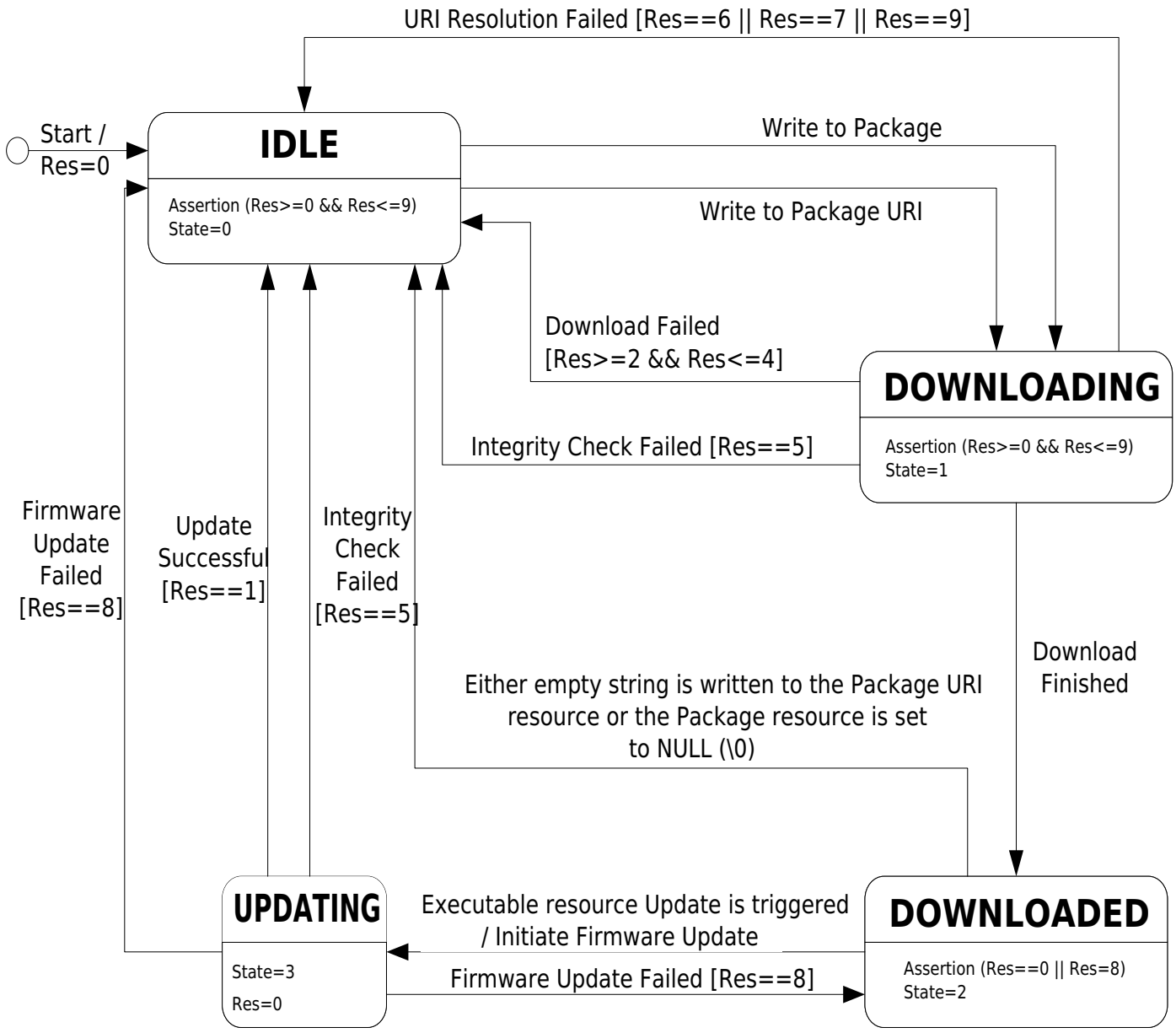


Figure: E.6.1-1 Firmware Update Mechanisms

E.6.2 Examples

The example depicted in Figure: E.6.2-1 Example of a LwM2M Server pushing a firmware image to a LwM2M client 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.

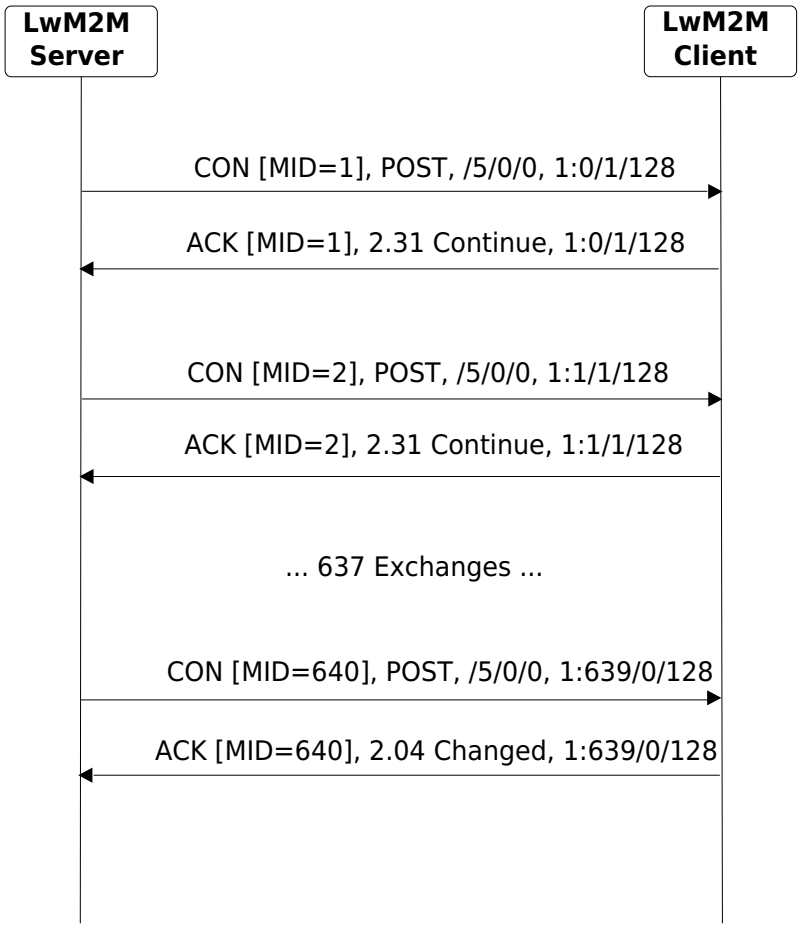


Figure: E.6.2-1 Example of a LwM2M Server pushing a firmware image to a LwM2M client

The second example shown in Figure: E.6.2-2 Example of a client fetching a firmware image 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: E.6.2-2 Example of a client fetching a firmware image and not the initial configuration of the Package URI.

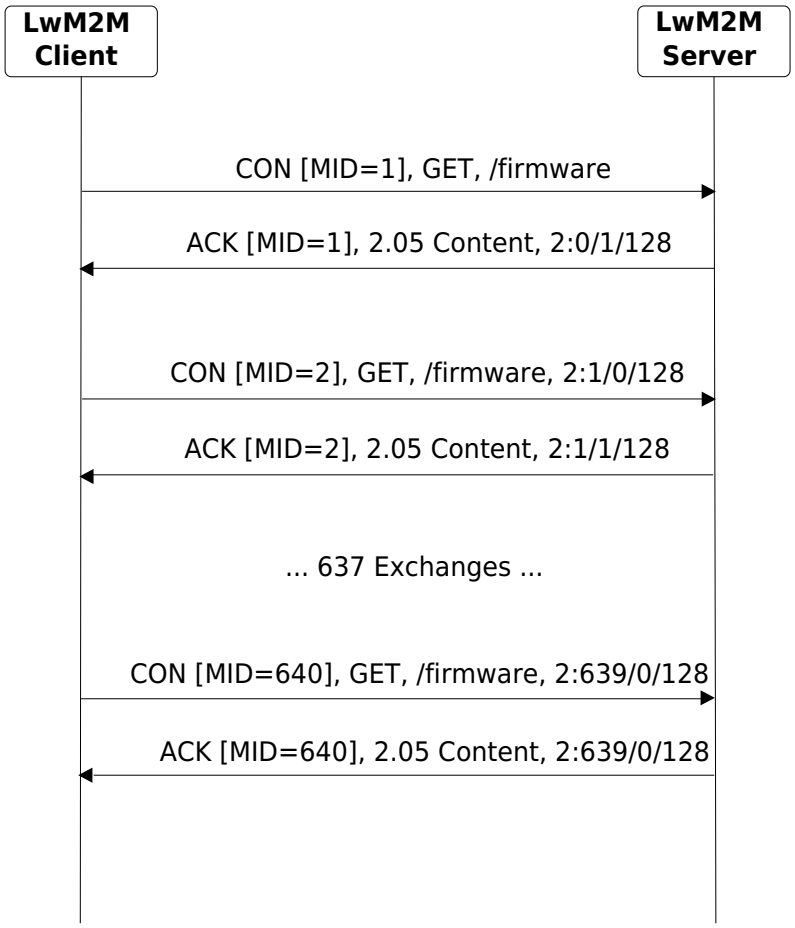


Figure: E.6.2-2 Example of a client fetching a firmware image

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

This LwM2M Object provides a range of location telemetry related information which can be queried by the LwM2M Server.

Object definition

Name	Object ID	Object Version	LWM2M Version
Location	6	1.0	1.0
Object URN		Instances	Mandatory

urn:oma:lwm2m:oma:6	Single	Optional

Table: E.7-1 LwM2M Object: Location object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Latitude	R	Single	Mandatory	Float		lat	The decimal notation of latitude, e.g. -43.5723 [World Geodetic System 1984].
1	Longitude	R	Single	Mandatory	Float		lon	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 this resource indicates the radius of a circular area in meters. The circular area is used to describe uncertainty about a point for coordinates in a two-dimensional coordinate reference systems (CRS). The center point of a circular area is specified by using the Latitude and the Longitude Resources.
4	Velocity	R	Single	Optional	Opaque			The velocity of the LwM2M Client, as 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		m/s	Speed is the time rate of change in position of a LwM2M Client without regard for direction: the scalar component of velocity.

Table: E.7-2 LwM2M Object: Location Resource definitions

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	Object Version	LWM2M Version
Connectivity Statistics	7	1.0	1.0
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:oma:7		Single	Optional

Table: E.8-1 LwM2M Object: Connectivity Statistics object definition

Table: E.8-1 LwM2M Object: Connectivity Statistics object definition

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			Indicate the total amount of data (IP / non-IP) transmitted during the collection period expressed in kilobytes.
3	Rx Data	R	Single	Optional	Integer			Indicate the total amount of data (IP / non-IP) received during the collection period expressed in kilobytes.
4	Max Message Size	R	Single	Optional	Integer		B	The maximum IP message size that is used during the collection period.
5	Average Message Size	R	Single	Optional	Integer		B	The average IP 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. Note:When reporting the Tx Data or Rx Data, the LwM2M Client reports the total KB transmitted/received over IP bearer(s), including all protocol header bytes up to and including the IP header. This does not include lower level retransmissions/optimizations (e.g. RAN, header compression) or SMS messages.
7	Stop	E	Single	Mandatory				Stop collecting information, but do not reset resources 0-5.
8	Collection Period	RW	Single	Optional	Integer		s	The default collection period in seconds. The value 0 indicates that the collection period is not set.

Table: E.8-2 LwM2M Object: Connectivity Statistics Resource definitions

Note: When reporting the Tx Data or Rx Data, the LwM2M Client reports the total KB transmitted/received over IP bearer(s), including all protocol header bytes up to and including the IP header. This does not include lower level retransmissions/optimizations (e.g. RAN, header compression) or SMS messages.

E.9 LwM2M Object: OSCORE

Description

This LwM2M Object provides the keying material and related information of a LwM2M Client appropriate to access a specified LwM2M Server using OSCORE. One Object Instance MAY 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. Instances of this Object are linked from Instances of Object 0 using the OSCORE Security Mode Resource of Object 0. Instances of this Object MUST NOT be linked from more than one Instance of Object 0.

Object definition

Name	Object ID	Object Version	LWM2M Version
LWM2M OSCORE	21	1.0	1.1
Object URN		Instances	Mandatory
urn:oma:lwm2m:oma:21		Multiple	Optional

Table: E.9-1 LwM2M Object: LWM2M OSCORE object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	OSCORE Master Secret		Single	Mandatory	String			This resource MUST be used to store the pre-shared key used in LwM2M Client and LwM2M Server/Bootstrap-Server, called the Master Secret.
1	OSCORE Sender ID		Single	Mandatory	String			This resource MUST store an OSCORE identifier for the LwM2M Client called the Sender ID.
2	OSCORE Recipient ID		Single	Mandatory	String			This resource MUST store an OSCORE identifier for the LwM2M Client called the Recipient ID.
3	OSCORE AEAD Algorithm		Single	Optional	Integer			This resource MUST be used to store the encoding of the AEAD Algorithm as defined in Table 10 of RFC 8152. The AEAD is used by OSCORE for encryption and integrity protection of CoAP message fields.
4	OSCORE HMAC Algorithm		Single	Optional	Integer			This resource MUST be used to store the encoding of the HMAC Algorithm used in the HKDF. The encoding of HMAC algorithms are defined in Table 7 of RFC 8152. The HKDF is used to derive the security context used by OSCORE.
5	OSCORE Master Salt		Single	Optional	String			This resource MUST be used to store a non-secret random value called the Master Salt. The Master Salt is used to derive the security context used by OSCORE.

Table: E.9-2 LwM2M Object: LWM2M OSCORE Resource definitions

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 are used throughout this specification in examples. The example client uses no Endpoint Client Name at the LwM2M messaging layer since the underlying security mechanisms already provide an authenticated identity.

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: F.-1 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		f81d4fae-7dec-11d0-a765-00a0c91e6bf6	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: F.-2 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: F.-3 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		3059301306072a8648ce 3d020106082a8648ce3d 030107034200 04a09dcb 1bc739e7f19afa9adcb1 944968bfba73efcec29b 50486eb44d1b29c19344 9970a3736830cb2bd5d7 ec05d2bd1fc07b6df5e4 8b54ce77a6a7229c3a91 c2	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		3059301306072a8648ce 3d020106082a8648ce3d 0301070342000482c773 f378d30c2783a48eb811 b96cf6a90694a8564f1e 7f6d366152f11698950f 6f7c40cda585334f8377 50a102f5bf25508ceb9e 55dfc0f12a455199a4c9 60	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		307702010104209f352d a16495748e146fcb5370 b8e96d292ced5567a8fa e55a22bb67d91651b8a0 0a06082a8648ce3d0301 07a144034200004a09dcb 1bc739e7f19afa9adcb1 944968bfba73efcec29b 50486eb44d1b29c19344 9970a3736830cb2bd5d7 ec05d2bd1fc07b6df5e4 8b54ce77a6a7229c3a91 c2	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: F.-4 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
    :    }
    :  }
```

The example above has been created as follows, as illustrated using the Server Public Key resource example. If the hex sequence of the Server Public Key resource is pasted into a text file with the name example.hex. Here is the content of example.hex:
'3059301306072a8648ce3d020106082a8648ce3d03010703420004 82c773f378d30c2783a48eb811b96cf6a90694a8564f1e7f6d366152f11698950f6f7c40cda585334f837750a102f5bf25508ceb9e55dfc0f12a455199a4c960'

Then, the hex data needs to be convered into a binary using the Linux xxd command:

```
xxd -r -p example.hex >public-key.pub
```

Finally, the binary needs to be interpreted as a public key using the Linux dumpasn1 command:

```
dumpasn1 public-key.pub
```

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: F.-5 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		U	UDP binding preference

Table: F.-6 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: F.-7 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: F.-8 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: F.-9 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: F.-10 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: F.-11 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: F.-12 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
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: F.-13 Connectivity Monitoring Object Instance

Appendix G. Storage of LwM2M Bootstrap Information on the Smartcard (Normative)

This appendix aims at specifying the storage of the LwM2M Bootstrap Information in a PKCS#15 file structure within an UICC Smartcard platform [ETSI TS 102.221].

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 [PKCS#15] specification defines a set of files.

```
MF (3F00)
|-EF DIR (2F00) --> reference PKCS\#15 Application & DF PKCS\#15
|-DF PKCS-15
|-ODF          --> ref to DODF          (Default:5031)
|-DODF         --> ref EF LwM2M Bootstrap (Default:6030)
|-EF LwM2M Bootstrap --> Contains LwM2M Bootstrap Data
```

Within the PKCS#15 application, the starting point to access the PKCS#15 files is the Object Directory File (ODF). The EF (ODF) contains pointers to other files,each one containing a directory over PKCS#15 objects of a particular class. (authentication objects, data objects, keys, certificates…). For the purpose of LwM2M Bootstrap capability, the EF(ODF) MUST contain an EF Record describing a DODF(Data Object Directory Files) in which is found the reference pointing to the LwM2M Bootstrap data.

- The EF (ODF) is described in G.3.2 Object Directory File, EF(ODF) and [PKCS#15].
- The EF (DODF-bootstrap) is described in G.3.3 Data Object Directory File, EF(DODF-bootstrap) and [PKCS#15].
- The EF (LwM2M Bootstrap data) is described in G.3.4 EF (LwM2M_Bootstrap).

G.2 Bootstrap Information on UICC

G.2.1 Access to the file structure

The selection of the PKCS#15 file structure or application is not within the scope of the specification and can be managed in different ways by the devices (Direct File Access using PKCS#15 AID , Indirect Access using EF Dir information).

However, the following sequence is a recommended way to perform the selection of the PKCS#15 application:

1. With only one PKCS#15 application present in the UICC, the device may send a SELECT command with the PKCS#15 AID (A0 00 00 00 63 50 4B 43 53 2D 31 35) as parameter (Direct Access). If the selection is successful, the device can start reading PKCS#15 files (ODF, DODF ..).
2. If the previous selection fails (or if it might have several PKCS#15 applications present in UICC), the device sends SELECT commands to access EFdir to locate an entry with the PKCS#15 AID. If only one matching entry is found, the device MUST select the PKCS#15 DF path from that entry, and then can start reading PKCS#15 files (ODF, DODF…). If several entries are matching, the device MUST look for the entry containing the OID related to LwM2M "2.23.43.9" see G.3 Files Description) and MUST then select the associated DODF where it will find the path for the

searched EF(LwM2M Bootstrap).

G.2.2 Files Overview (example)

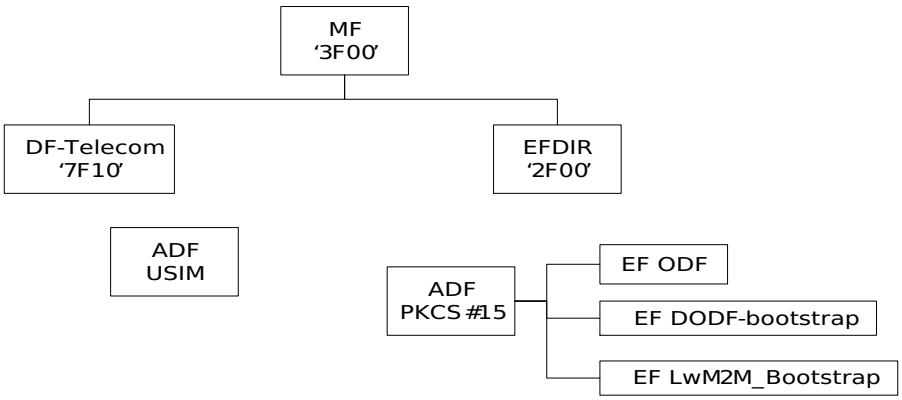


Figure: G.2.2-1 Example of a UICC File Structure embedding a specific PKCS\#15 file structure containing LwM2M Bootstrap data location

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 [PKCS#15].

For the M2M context, the flags mentioned above can be unset to prevent cardholder verification step through pin code. When more control is needed, the Secure Channel SHOULD be used Appendix H. Secure channel between Smartcard and LwM2M Device Storage for secure Bootstrap Data provisioning (Normative).

G.2.5 Requirements on the Device

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

- Select PKCS#15 file structure as recommended in G.2.1 Access to the file structure.
- if needed (single PKCS#15 application case) Read EF(ODF) to locate the EF(DODF-bootstrap),
- Read EF(DODF) and look for the OMA-LwM2M OID to locate the file containing the LwM2M_Bootstrap data,
- Read the LwM2M_Bootstrap file

G.3 Files Description

All PKCS#15 files defined, are binary files as specified in [ETSI TS 102.221] and encoded in DER Format [PKCS#15]. These files are read and updated using UICC Commands related to the application they belong to.

G.3.1 EF(DIR) – optional

When the optional EF (DIR) is used to locate the PKCS#15 files structure, one of its logical record MUST be of the following ASN.1 type with the specified [PKCS#15] tags :

- Application template Tag 0x61
- Application Identifier Tag 0x4F
- Path Tag 0x51
- Application label Tag 0x50
- data Object Tag 0x73 (Multiple PKCS#15 Application case)

```
DIREcord ::= [APPLICATION 1]  SEQUENCE {
  aid   [APPLICATION 15] OCTET STRING,
  label [APPLICATION 16] UTF8STRING OPTIONAL
  path  [APPLICATION 17] OCTET STRING
  ddo   [APPLICATION 19] DDO OPTIONAL // used in multiple PKCS\#15 Applications case and containing
                              // OID & associated DODF path (see [Requirements on the Device]() & \[PKCS#15\])
}

Coding  example :
aid PKCS#15  =  A0 00 00 00 63 50 4B 43 53 2D 31 35    (Tag : 0x4F)
label              = “BOOTSTRAP”                  (Tag : 0x50)
path               = 3F00/7F50                   (Tag : 0x51)

      61 22
         4F 0C A0 00 00 00 63 50 4B 43 53 2D 31 35
         50 08  42  4F 4F 54 53 54 52 41 50
         51 04  3F  00 7F 50
```

G.3.2 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 (Keys, Authentication, Data ..). The EF(ODF) File ID is specified in [PKCS#15](0x5031). 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 : Binary coding example (DER format) A7 06 30 04 04 02 64 30 (reference DODF = 0x6430)		
See [PKCS#15]		

Table: G.3.2-1 Object Directory File, EF ODF

G.3.3 Data Object Directory File, EF(DODF-bootstrap)

This Data Object Directory File used as the entry point to the LwM2M Bootstrap data, MUST contains an oidDO entry [PKCS#15] with an OID that MUST be in the OMA-LwM2M scope.

The registered OID for this specification is: {joint-isu-itu-t(2) international-organizations(23) oma(43) oma-lwm2m(9) lwm2m_bootstrap(1)} (2.23.43.9.1).

According to [PKCS#15], the DODF MUST include a DataType with the oidDO entry. The DataType oidDO entry is defined as PKCS15Object {CommonDataObjectAttributes, NULL, OidDO}.

- The CommonDataObjectAttributes structure of the DODF DataType oidDO entry MAY contain an applicationName or an applicationOID. The applicationName and the applicationOID are informative in this specification (may be omitted).
- The OidDO structure of the DODF DataType oidDO entry MUST be present and MUST contain an id value set to the registered OID defined above and a value that is a path structure to the EF (LwM2M Bootstrap data).

This path structure is defined with the following ASN.1 syntax [PKCS#15]: Path ::= SEQUENCE { path OCTET STRING, index INTEGER (0..65535) OPTIONAL, length [0] INTEGER (0..65535) OPTIONAL } (WITH COMPONENTS {..., index PRESENT, length PRESENT} | WITH COMPONENTS {..., index ABSENT, length ABSENT}) with "path" being the path to the EF (LwM2M Bootstrap data)

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 G.2 Bootstrap Information on UICC) UPDATE ADM INVALIDATE ADM REHABILITATE ADM		
Description : : Binary coding example (DER format) Label = "LwM2M Bootstrap" HEX : 4C774D324D20426F6F747374726170 OID OMA-LwM2M Bootstrap "2.23.43.9.1" (HEX DER) 06 04 67 2B 09 01 Path = 0x6432 A1 27 30 00 30 11 0C 0F 4C 77 4D 32 4D 20 42 6F 6F 74 73 74 72 61 70 A1 10 30 0E 06 06 06 04 67 2B 09 01 30 04 04 02 64 32		
See hereafter and [PKCS#15]		

Table: G.3.3-1 Bootstrap Data Object Directory File, EF DODF-bootstrap

G.3.4 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 G.2 Bootstrap Information on UICC) UPDATE ADM INVALIDATE ADM REHABILITATE ADM
Description
Contains Bootstrap data (encapsulated LwM2M Objects)

Table: G.3.4-1 EF LwM2M_Bootstrap

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 (7.4.4. TLV), 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). Data are represented in network byte order (big endian).

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 7.4.4. TLV

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 between the Smartcard and the LwM2M Client. When required this Secure Channel MUST follow the following recommendations provided by [GLOBALPLATFORM] [GP SCP03] which are illustrated below. The Bootstrap information will be retrieved from Smartcard as described in Appendix F. Example LwM2M Client (Informative) of this document with the modification the usage of introducing a secured transfer between the Smartcard and the Device: instead of using PKCS#15 application, a specific OMA-LwM2M Bootstrap application MUST be selected in the Smartcard (SELECT command). The File Structure where the LwM2M Bootstrap Data is located, MUST only be accessible if the Mutual Authentication of the GP SCP03 process has been satisfied.

The AID (0xA0000004120002000000000) of the OMA-LwM2M Bootstrap Application is composed of the OMA RID code (0xA000000412), and the OMA PIX code containing the LwM2M Application code (0x0002) and a specific code used by the application; this specific code is relative to the Bootstrap data retrieval (00 00 00 00 00)

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 secured transfer are the following and are illustrated below Figure: H.-1 Bootstrap Information transfer from Smartcard to LwM2M Device using Secure Channel according to [GLOBALPLATFORM] [GP SCP03] [GP AMD_A]:

- The OMA-LwM2M Bootstrap application used for transferring the Bootstrap information is selected
- Secure Channel (mutual authentication) is established
- PKCS#15 flow as described in Appendix F. Example LwM2M Client (Informative) takes place for selecting and transferring the Bootstrap file from Smartcard to the Device: the sensitive Bootstrap data are transferred encrypted using the GP SCP03 Secure Channel. When the Mutual Authentication is not satisfied, access to the file structure will return an error code 6982 to the Device (Security status not satisfied).

As a general recommendation, in the process for retrieving the Bootstrap data from the Smartcard, the Device SHOULD firstly try to select the OMA-LwM2M Bootstrap application to activate the [GP SCP03] Secure Channel procedure; if such an application is not present, the Device SHOULD then try to get this Bootstrap Data by selecting the PCKS#15 application as described in Appendix G. Storage of LwM2M Bootstrap Information on the Smartcard (Normative).

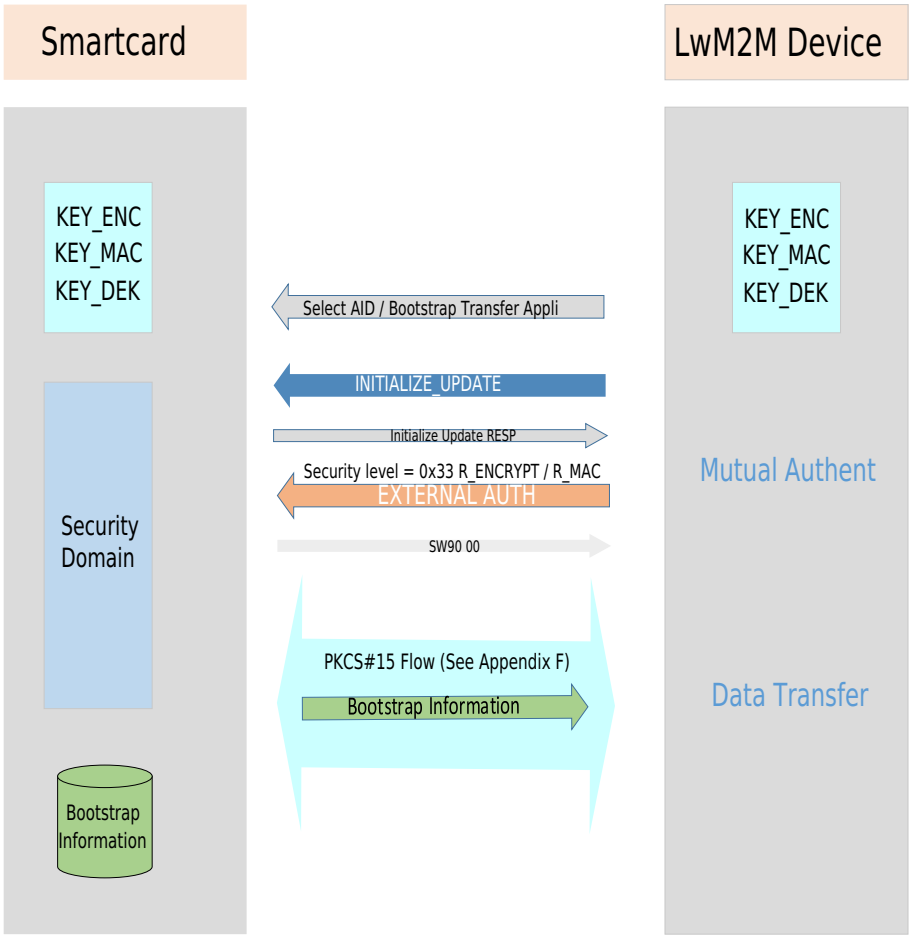


Figure: H.-1 Bootstrap Information transfer from Smartcard to LwM2M Device using Secure Channel according to [GLOBALPLATFORM] [GP SCP03] [GP AMD_A]

The OMA-LwM2M Bootstrap application (AID : (0xA00000041200020000000000) MUST support the following commands : As specified in [GLOBALPLATFORM]:

- INITIALIZE UPDATE
- EXTERNAL AUTHENTICATE with P1=0x33 (C-DECRYPTION, R-ENCRYPTION, C-MAC and R-MAC)

As defined in [ETSI TS 102.221]:

- SELECT with P1=0x00 (select by file ID)
- READ BINARY Any other command is optional.

I Media types

This enabler doesn't request any new media types from IANA.

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

application/vnd.oma.lwm2m+tlv has been registered with IANA in version 1.0.

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

application/vnd.oma.lwm2m+json has been registered with IANA in version 1.0.

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. LwM2M Schema.

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

1. the "LwM2MVersion" element indicates the minimum version of the LwM2M Enabler supporting that Object,
2. the "ObjectVersion" element indicates the version of the Object as defined in Section 7.2. Object Versioning of 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

Appendix L. Example of Trigger Message from Server (Informative)

Example WAP Push over SMS containing the trigger information:

Binary value	Meaning	Description
06	User-Data-Header (UDHL) Length = 6 bytes	WDP layer (start WDP headers).
05	UDH IE identifier: Port numbers	
04	UDH port number IE length	
0B	Destination port (high)	Port number 2948
84	Destination port (low)	
C0	Originating port (high)	Port number chosen by sender
02	Originating port (low)	WDP layer (end WDP headers)
01	Transaction ID / Push ID	WSP layer (start WSP headers)
06	PDU type (push)	
03	Headerslength (content type+headers)	
C4	Content type code	MIME-Type
AF	X-WAP-Application-ID	
9A	Id for urn: x-wap-application:lwm2m.dm	WSP layer (end WSP headers)
{14-bytes}	112-bit clear value	Clear CoAP Text

Table: L.-1 Example WAP Push over SMS containing the trigger information

CoAP raw bytes	Description

44 02 b6 0b 21 61 fb 63 b1 31 01 30 01 38	PDU length:14 CoAP Version: 1 Message Type: Confirmable Token length: 4 Code: 0.02 POST Message ID: 34488 Token length: 4 Value: 0x301e65ca OPTION (0/3) Option number (delta): 11(11) Name: URI_PATH Value length: 1 Value: "1" OPTION (1/3) Option number (delta): 11(0) Name: URI_PATH Value length: 1 Value: "0" OPTION (2/3) Option number (delta): 11(0) Name: URI_PATH Value length: 1 Value: "8" No payload.

Table: L.-2 Example WAP Push over SMS containing the trigger information

Appendix M. CBOR Example (Informative)

The example request to Device Object of the LwM2M example client (Read /3/0) is shown below using the CBOR format with comments.

```
90                      # array(16)
  A3                    # map(3)
    21                  # negative(1)
    65                  # text(5)
      2F332F302F            # "/3/0/"
    00                  # unsigned(0)
    61                  # text(1)
      30                # "0"
    03                  # unsigned(3)
    74                  # text(20)
      4F70656E204D6F62696C6520416C6C69616E6365 # "Open Mobile Alliance"
  A2                    # map(2)
    00                  # unsigned(0)
    61                  # text(1)
      31                # "1"
    03                  # unsigned(3)
    76                  # text(22)
      4C696768747765696768742 04D324D20436C69656E74 # "Lightweight M2M Client"
  A2                    # map(2)
    00                  # unsigned(0)
    61                  # text(1)
      32                # "2"
    03                  # unsigned(3)
    69                  # text(9)
      333435303030313233        # "345000123"
  A2                    # map(2)
    00                  # unsigned(0)
    61                  # text(1)
      33                # "3"
    03                  # unsigned(3)
    63                  # text(3)
      312E30              # "1.0"
  A2                    # map(2)
    00                  # unsigned(0)
    63                  # text(3)
      362F30              # "6/0"
    02                  # unsigned(2)
    01                  # unsigned(1)
  A2                    # map(2)
    00                  # unsigned(0)
    63                  # text(3)
      362F31              # "6/1"
    02                  # unsigned(2)
    05                  # unsigned(5)
  A2                    # map(2)
    00                  # unsigned(0)
    63                  # text(3)
      372F30              # "7/0"
    02                  # unsigned(2)
```

```
   19 0ED8                  # unsigned(3800)
A2                          # map(2)
   00                       # unsigned(0)
   63                       # text(3)
      372F31                # "7/1"
   02                       # unsigned(2)
   19 1388                  # unsigned(5000)
A2                          # map(2)
   00                       # unsigned(0)
   63                       # text(3)
      382F30                # "8/0"
   02                       # unsigned(2)
   18 7D                    # unsigned(125)
A2                          # map(2)
   00                       # unsigned(0)
   63                       # text(3)
      382F31                # "8/1"
   02                       # unsigned(2)
   19 0384                  # unsigned(900)
A2                          # map(2)
   00                       # unsigned(0)
   61                       # text(1)
      39                    # "9"
   02                       # unsigned(2)
   18 64                    # unsigned(100)
A2                          # map(2)
   00                       # unsigned(0)
   62                       # text(2)
      3130                  # "10"
   02                       # unsigned(2)
   0F                       # unsigned(15)
A2                          # map(2)
   00                       # unsigned(0)
   64                       # text(4)
      31312F30              # "11/0"
   02                       # unsigned(2)
   00                       # unsigned(0)
A2                          # map(2)
   00                       # unsigned(0)
   62                       # text(2)
      3133                  # "13"
   02                       # unsigned(2)
   1A 5182428F              # unsigned(1367491215)
A2                          # map(2)
   00                       # unsigned(0)
   62                       # text(2)
      3134                  # "14"
   03                       # unsigned(3)
   66                       # text(6)
      2B30323A3030          # "+02:00"
A2                          # map(2)
   00                       # unsigned(0)
   62                       # text(2)
      3136                  # "16"
   03                       # unsigned(3)
   61                       # text(1)
```

```
55                  # "U"
```