

Lightweight Machine to Machine Technical Specification: Core

Draft Version: 1.1 – 08 Jan 2018

Open Mobile Alliance

OMA-TS-LightweightM2M_Core-V1_1

Merged: 28 Feb 2018 09:32:00 *rev: 047f80f*

Use of this document is subject to all of the terms and conditions of the Use Agreement located at <http://www.openmobilealliance.org/UseAgreement.html>.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this

document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the “OMA IPR Declarations” list at <http://www.openmobilealliance.org/ipr.html>. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR’S REPRESENTED ON THE “OMA IPR DECLARATIONS” LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

© 2018 Open Mobile Alliance All Rights Reserved. Used with the permission of the Open Mobile Alliance under the terms set forth above.

Table of Contents

[1. Scope](#)

[1.1. V1_0_x](#)

[1.2. V1_1_x](#)

[2. References](#)

[2.1. Normative References](#)

[2.2. Informative References](#)

[3. Terminology and Conventions](#)

[3.1. Conventions](#)

[3.2. Definitions](#)

[3.3. Abbreviations](#)

[4. Introduction](#)

[4.1. Version 1.0](#)

[5. Interfaces](#)

[5.1. Attributes](#)

[5.1.1. Attributes Definitions and Rules](#)

[5.1.2. Attributes Classification](#)

[5.2. Bootstrap Interface](#)

[5.2.1. LwM2M Bootstrap-Server](#)

[5.2.2. Bootstrap Information](#)

[5.2.3. Bootstrap Modes](#)

[5.2.3.1. Factory Bootstrap](#)

[5.2.3.2. Bootstrap from Smartcard](#)

[5.2.3.3. Client Initiated Bootstrap](#)

[5.2.3.4. Server Initiated Bootstrap](#)

[5.2.4. Bootstrap Sequence](#)

[5.2.5. Bootstrap Security](#)

[5.2.6. Bootstrap and Configuration Consistency](#)

[5.2.7. Bootstrap Commands](#)

[5.2.7.1. BOOTSTRAP-REQUEST](#)

[5.2.7.2. BOOTSTRAP-FINISH](#)

[5.2.7.3. BOOTSTRAP DISCOVER](#)

[5.2.7.4. 5.2.7.4 BOOTSTRAP READ](#)

[5.2.7.5. BOOTSTRAP WRITE](#)

[5.2.7.6. BOOTSTRAP DELETE](#)

[5.3. Client Registration Interface](#)

[5.3.1. Register](#)

[5.3.1.1. Behaviour with Current Transport Binding and Mode](#)

[5.3.2. Update](#)

[5.3.3. De-register](#)

[5.4. Device Management & Service Enablement Interface](#)

[5.4.1. Read](#)

[5.4.2. Discover](#)

[5.4.3. Write](#)

[5.4.4. Write-Attributes](#)

[5.4.5. Execute](#)

[5.4.6. Create](#)

[5.4.7. Delete](#)

[5.4.8. Read-Composite](#)

[5.4.9. Write-Composite](#)

[5.5. Information Reporting Interface](#)

[5.5.1. Observe](#)

[5.5.2. Notify](#)

[5.5.3. Cancel Observation](#)

[5.5.4. Observe-Composite](#)

[6. Identifiers and Resources](#)

[6.1. Resource Model](#)

[6.2. Object Versioning](#)

[6.2.1. General Policy](#)

[6.2.2. Object Version format](#)

[6.2.3. Object Definition and Object Version Usage](#)

[6.3. Identifiers](#)

[6.3.1. Endpoint Client Name](#)[6.3.2. Reusable Resources](#)[6.4. Data Formats for Transferring Resource Information](#)[6.4.1. Plain Text](#)[6.4.2. Opaque](#)[6.4.3. TLV](#)[6.4.3.1. Single Object Instance Request Example](#)[6.4.3.2. Multiple Object Instance Request Examples](#)[6.4.3.3. Example of Request on an Object Instance containing an Object Link Resource](#)[6.4.4. JSON](#)[6.4.5. CBOR](#)[7. Access Control](#)[7.1. Access Control Object](#)[7.1.1. Access Control Object overview](#)[7.1.2. Access Control Object Management](#)[7.1.2.1. Access Control on Object](#)[7.1.2.2. Access Control on Object Instance](#)[7.1.3. LwM2M Server Context Switch](#)[7.2. Authorization](#)[7.2.1. Obtaining Access Right](#)[7.2.2. Operation on Resource](#)[7.2.3. Operation on Object Instance](#)[7.2.4. Operation on Object](#)[7.2.5. Notify Operation Consideration](#)[Appendix A. Change History \(Informative\)](#)[A.1 Approved Version History](#)[A.2 Draft Version History](#)[Appendix B. Static Conformance Requirements \(Normative\)](#)[B.1 SCR for LWM2M Client](#)[B.1.1 Bootstrap Interface](#)[B.1.2 Client Registration](#)[B.1.3 Device Management and Service Enablement Interface](#)[B.1.4 Information Reporting](#)

[B.1.5 Data Format](#)

[B.1.6 Security](#)

[B.1.7 Mechanism](#)

[B.1.8 Objects](#)

[B.2 SCR for LwM2M Server](#)

[B.2.1 Bootstrap Interface](#)

[B.2.2 Client Registration](#)

[B.2.3 Device Management and Service Enablement Interface](#)

[B.2.4 Information Reporting](#)

[B.2.5 Data Format](#)

[B.2.6 Security](#)

[B.2.7 Mechanism](#)

[B.2.8 Objects](#)

[Appendix C. Data Types \(Normative\)](#)

[Appendix D. LwM2M Object Template and Guidelines \(Normative\)](#)

[D.1 Object Template](#)

[D.2 Open Mobile Naming Authority \(OMNA\) Guidelines](#)

[D.2.1 Object Registry](#)

[D.2.2 Resource Registry](#)

[Appendix E. LwM2M Objects defined by OMA \(Normative\)](#)

[E.1 LwM2M Object: LwM2M Security](#)

[E.2 LwM2M Object: LwM2M Server](#)

[E.3 LwM2M Object: Access Control](#)

[E.4 LwM2M Object: Device](#)

[E.5 LwM2M Object: Connectivity Monitoring](#)

[E.6 LwM2M Object: Firmware Update](#)

[E.6.1 Firmware Update State Machine](#)

[E.6.2 Examples](#)

[E.6.3 Firmware Update Consideration](#)

[E.7 LwM2M Object: Location](#)

[E.8 LwM2M Object: Connectivity Statistics](#)

[Appendix F. Example LwM2M Client \(Informative\)](#)

[Appendix G. Storage of LwM2M Bootstrap Information on the Smartcard \(Normative\)](#)

[G.1 File structure](#)

[G.2 Bootstrap Information on UICC](#)

[G.2.1 Access to the file structure](#)

[G.2.2 Files Overview \(example\)](#)

[G.2.3 Access Method](#)

[G.2.4 Access Conditions](#)

[G.2.5 Requirements on the Device](#)

[G.3 Files Description](#)

[G.3.1 EF\(DIR\) – optional](#)

[G.3.2 Object Directory File, EF\(ODF\)](#)

[G.3.3 Data Object Directory File, EF\(DODF-bootstrap\)](#)

[G.3.4 EF \(LwM2M_Bootstrap\)](#)

[Appendix H. Secure channel between Smartcard and LwM2M Device Storage for secure Bootstrap Data provisioning \(Normative\)](#)

[Appendix I. Media types](#)

[I.1 Media-Type application/vnd.oma.lwm2m+tlv Registration](#)

[I.2 Media-Type application/vnd.oma.lwm2m+json Registration](#)

[Appendix J. LwM2M Schema and Object Definition File \(Informative\)](#)

[Appendix K. LwM2M Schema](#)

[Appendix L. Example of Trigger Message from Server \(Informative\)](#)

[Appendix M. LWM2M over NB-IoT](#)

[M.1 Introduction](#)

[M.2 NIDD via PtP IP SGi tunnel](#)

[M.3 NIDD via SCEF](#)

[M.4 NAS Transport](#)

[M.5 Large data transport with NB-IoT](#)

[M.6 Message buffering](#)

[M.7 NB-IoT transport configuration options](#)

[M.8 Timer considerations](#)

[M.8.1 Introduction](#)

[M.8.2 3GPP Parameters](#)

[M.8.3 LwM2M Parameters](#)

[M.8.4 Interactions between parameters](#)

[M.8.5 Timer implementation options](#)

Table of Figures

[Figure: 4.-1 The overall architecture of the LwM2M Enabler](#)

[Figure: 5.-1 Bootstrap](#)

[Figure: 5.-2 Client Registration](#)

[Figure: 5.-3 Device Management and Service Enablement](#)

[Figure: 5.-4 Information Reporting](#)

[Figure: 5.2.3.3.-1 Procedure of Client Initiated Bootstrap](#)

[Figure: 5.2.3.4.-1 Procedure of Server Initiated Bootstrap initiated by an authorized LwM2M Server](#)

[Figure: 5.3.-1 Client Registration Interface example flows](#)

[Figure: 5.3.2.-1 Client Registration Update example flows #1](#)

[Figure: 5.3.2.-2 Client Registration Update example flows #2](#)

[Figure: 5.4.-1 Example flows of Device Management & Service Enablement Ineterface](#)

[Figure: 5.4.5.-1 ABNF syntax](#)

[Figure: 5.5.-1 Example flow for Information Reporting Interface for the RSSI Resource of the Connectivity Monitoring Object of the example client \(Appendix E\)](#)

[Figure: 5.5.2.-1 Example of Minimum and Maximum periods in an Observation](#)

[Figure: 6.1.-1 Relationship between LwM2M Client, Object, and Resources](#)

[Figure: 6.1.-2 Example of Supported operations and Associated Access Control Object Instance](#)

[Figure: 6.4.3.-1 TLV nesting](#)

[Figure: 7.1.2.2.-1 Illustration of the relations between the LwM2M Access Control Object and the other LwM2M Objects](#)

[Figure: C.-1 Object link Resource simple illustration](#)

[Figure: E.6.1-1 Firmware Update Mechanisms](#)

[Figure: E.6.2-1 Example of a LwM2M Server pushing a firmware image to a LwM2M client](#)

[Figure: E.6.2-2 Example of a client fetching a firmware image](#)

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

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

[Figure: M.1-1 3GPP NB-IoT architecture as in \[3GPP-TR_23.720\] and the LWM2M protocol stack](#)

Table of Tables

[Table: 2.1.-1 Normative References](#)

[Table: 2.2.-1 Informative References](#)

[Table: 3.2.-1 Definitions](#)

[Table: 3.3.-1 Abbreviations](#)

[Table: 5.-1 Relationship of operations and interfaces](#)

[Table: 5.1.1.-1 Attribute Characteristics](#)

[Table: 5.1.2.-1 <PROPERTIES> Class Attributes](#)

[Table: 5.1.2.-2 <NOTIFICATION> class Attributes](#)

[Table: 5.2.2.-1 Bootstrap Information List](#)

[Table: 5.2.7.1.-1 Bootstrap-Request operation parameter](#)

[Table: 5.2.7.2.-1 Bootstrap-Finish operation parameter](#)

[Table: 5.2.7.3.-1 Bootstrap Discover parameters](#)

[Table: 5.2.7.5.-1 Bootstrap “Write” operation parameters](#)

[Table: 5.2.7.6.-1 Delete operation parameters](#)

[Table: 5.3.1.-1 Registration parameters](#)

[Table: 5.3.1.1.-1 Behaviour with Current Transport Binding and Mode](#)

[Table: 5.3.2.-1 Update parameters](#)

[Table: 5.4.1.-1 Read parameters](#)

[Table: 5.4.2.-1 Discover parameters](#)

[Table: 5.4.3.-1 Write parameters](#)

[Table: 5.4.4.-1 Write-Attributes parameters](#)

[Table: 5.4.5.-1 Execute parameters](#)

[Table: 5.4.6.-1 Create parameters](#)

[Table: 5.4.7.-1 Delete parameters](#)

[Table: 5.5.1.-1 Observe parameters](#)

[Table: 5.5.2.-1 Notify parameters](#)

[Table: 6.2.3.-1 Object Version usage rules](#)

[Table: 6.3.-1 LwM2M Identifiers](#)

[Table: 6.3.1.-1 Endpoint Client Name](#)

[Table: 6.4.-1 IANA registered Media Type supported in LwM2M TS 1.0](#)

[Table: 6.4.3.-1 TLV format and description](#)

[Table: 6.4.3.1.-1 Single Object Instance Request Example](#)

[Table: 6.4.3.2.-1 Request on Single-Instance Object](#)

[Table: 6.4.3.2.-2 Request on Multiple-Instance Object having 2 instances](#)

[Table: 6.4.3.2.-3 Example, a request to the Server Object Instances of a LwM2M client is performed \(Read /1\)](#)

[Table: 6.4.3.3.-1 Example 1\) Request to Object 65 Instance 0: Read /65/0](#)

[Table: 6.4.3.3.-2 Example 2\) request to Object 66: Read /66: TLV payload will contain 2 Object Instances](#)

[Table: 6.4.4.-1 JSON format and description](#)

[Table: 6.4.4.-2 Return JSON payload from example request to Device Object of the LwM2M example client \(Read /3/0\)](#)

[Table: 6.4.4.-3 JSON payload from example notification about a Resource containing multiple historical representations of a Temperature Resource](#)

[Table: 6.4.4.-4 JSON payload returned from example request to Object 65 of the LwM2M example from Figure 19 \(Read /65/0\)](#)

[Table: 6.4.4.-5 JSON payload returned from example request to Device Object on Resource 0 of the LwM2M example client \(Read /3/0/0\)](#)

[Table: 6.4.4.-6 JSON payload in the server request to read manufacturer name, battery level and registration lifetime](#)

[Table: 6.4.4.-7 JSON payload in the client response to server request to read manufacturer name, battery level and registration lifetime](#)

[Table: 6.4.4.-8 JSON payload in the server Write-Composite to switch off /3311/0 and /3311/1, while dimming /3311/2](#)

[Table: 7.1.2.1.-1 Access Control on Object](#)

[Table: 7.1.2.2.-1 Access Control on Object Instance](#)

[Table: 7.2.-1 Authorization](#)

[Table: A.1-1 Approved Version History](#)

[Table: A.2-1 Draft Version History](#)

[Table: B.1.1-1 Bootstrap Interface](#)

[Table: B.1.2-1 Client Registration](#)

[Table: B.1.3-1 Device Management and Service Enablement Interface](#)

[Table: B.1.4-1 Information Reporting](#)

[Table: B.1.5-1 Data Format](#)

[Table: B.1.6-1 Security](#)

[Table: B.1.7-1 Mechanism](#)

[Table: B.1.8-1 Objects](#)

[Table: B.2.1-1 Bootstrap Interface](#)

[Table: B.2.2-1 Client Registration](#)

[Table: B.2.3-1 Device Management and Service Enablement Interface](#)

[Table: B.2.4-1 Information Reporting](#)

[Table: B.2.5-1 Data Format](#)

[Table: B.2.6-1 Security](#)

[Table: B.2.7-1 Mechanism](#)

[Table: B.2.8-1 Objects](#)

[Table: C.-1 Data Types](#)

[Table: D.1-1 Object definition](#)

[Table: D.1-2 Resource definition](#)

[Table: D.1-3 Executable Resource Arguments Definition](#)

[Table: D.1-4 Example Executable Resource Arguments Definition](#)

[Table: E.-1 LwM2M Objects defined by OMA LwM2M 1.0](#)

[Table: E.1-1 LwM2M Object: LwM2M Security object definition](#)

[Table: E.1-2 LwM2M Object: LwM2M Security resource definitions](#)

[Table: E.2-1 LwM2M Object: LwM2M Server Object definition](#)

[Table: E.2-2 LwM2M Object: LwM2M Server Resource definitions](#)

[Table: E.3-1 LwM2M Object: Access Control object definition](#)

[Table: E.3-2 LwM2M Object: Access Control resource definitions](#)

[Table: E.4-1 LwM2M Object: Device object definition](#)

[Table: E.4-2 LwM2M Object: Device resource definitions](#)

[Table: E.4-3 Battery Status](#)

[Table: E.5-1 LwM2M Object: Connectivity Monitoring object definition](#)

[Table: E.5-2 LwM2M Object: Connectivity Monitoring resource definitions](#)

[Table: E.6-1 LwM2M Object: Firmware Update object definition](#)

[Table: E.6-2 LwM2M Object: Firmware Update resource definitions](#)

[Table: E.7-1 LwM2M Object: Location object definition](#)

[Table: E.7-2 LwM2M Object: Location resource definitions](#)

[Table: E.8-1 LwM2M Object: Connectivity Statistics object definition](#)

[Table: E.8-2 LwM2M Object: Connectivity Statistics resource definitions](#)

[Table: F.-1 Object Instances of the example](#)

[Table: F.-2 LwM2M Security Object \[0\]](#)

[Table: F.-3 LwM2M Security Object \[1\]](#)

[Table: F.-4 LwM2M Security Object \[2\]](#)

[Table: F.-5 LwM2M Server Object \[0\]](#)

[Table: F.-6 LwM2M Server Object \[1\]](#)

[Table: F.-7 Access Control Object \[0\] \(for the LwM2M Server Object Instance 0\)](#)

[Table: F.-8 Access Control Object \[1\] \(for the LwM2M Server Object Instance 1\)](#)

[Table: F.-9 Access Control Object \[2\] \(for the Device Object Instance\)](#)

[Table: F.-10 Access Control Object \[3\] \(for the Connectivity Monitoring Object Instance\)](#)

[Table: F.-11 Access Control Object \[4\] \(for the Firmware Update Object\)](#)

[Table: F.-12 Device Object Instance](#)

[Table: F.-13 Connectivity Monitoring Object Instance](#)

[Table: G.3.2-1 Object Directory File, EF ODF](#)

[Table: G.3.3-1 Bootstrap Data Object Directory File, EF DODF-bootstrap](#)

[Table: G.3.4-1 EF LwM2M_Bootstrap](#)

[Table: L.-1 Example WAP Push over SMS containing the trigger information](#)

[Table: L.-2 Example WAP Push over SMS containing the trigger information](#)

[Table: M.4-1 'Release assistance indication' value and Recommended Use](#)

[Table: M.8.2-1 3GPP Parameters](#)

[Table: M.8.3-1 LwM2M Parameters](#)

1. Scope

1.1. V1_0_x

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

- Simple resource model with the core set of objects and resources defined in this specification. The full list of registered objects can be found at [OMNA](#).
- Operations for creation, update, deletion, and retrieval of resources.
- Asynchronous notifications of resource changes.
- Support for several serialization formats, namely TLV, JSON, CBOR, 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.

1.2. V1_1_x

2. References

2.1. Normative References

[LwM2M-TRANSPORT]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core Layer", (work in progress), Dec. 2017.
[3GPP-TS_23.003]	3GPP TS 23.003 "Numbering, addressing and identification"
[3GPP-TS_23.032]	3GPP TS 23.032 "Universal Geographical Area Description (GAD)"

[3GPP-TS_23.038]	3GPP TS 23.038 “Alphabets and language-specific information”
[3GPP-TS_23.040]	3GPP TS 23.040 “Technical realization of the Short Message Service (SMS)”
[3GPP-TS_24.008]	3GPP TS 24.008 “Mobile radio interface Layer 3 specification; Core network protocols; Stage 3”
[3GPP-TS_25.331]	3GPP TS 25.331 “Radio Resource Control (RRC); Protocol specification”
[3GPP-TS_31.111]	3GPP TS 31.111 “Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)”
[3GPP-TS_31.115]	3GPP TS 31.115 “Remote APDU Structure for (U)SIM Toolkit applications”
[CoAP]	Shelby, Z., Hartke, K., Bormann, C., and B. Frank, “The Constrained Application Protocol (CoAP)”, IETF RFC 7252 – June 2014
[CoAP_Blockwise]	C. Bormann, Z. Shelby, “Block-wise transfers in CoAP”, IETF RFC 7959.
[CoRE_Interface]	Z. Shelby, M. Vial, “CoRE Interfaces”, draft-ietf-core-interfaces-01, Nov 2013
[ETSI TS 102.221]	“Smart Cards; UICC-Terminal interface; Physical and logical characteristics”, (ETSI TS 102 221 release 11), URL:http://www.etsi.org/
[ETSI TS 102.223]	“Smart Cards; Card Applications Toolkit (CAT) (Release 11)” URL:http://www.etsi.org/
[ETSI TS 102.225]	ETSI TS 102 225 (V11.0.0): “Smart Cards; Secured packet structure for UICC based applications (Release 11)” URL:http://www.etsi.org/
[FLOAT]	IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008
[GLOBALPLATFORM]	GlobalPlatform v2.2.1 – January 2011 –
[GP SCP03]	GlobalPlatform Secure Channel Protocol 03 (SCP 03) Amendment D v1.1 Sept 2009
[IEEE 754-2008]	IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008

[IOPPROC]	“OMA Interoperability Policy and Process”, Version 1.13, Open Mobile Alliance™, OMA-IOP-Process-V1_13, URL:http://www.openmobilealliance.org/
[LwM2M-AD]	“Lightweight Machine to Machine Architecture”, Open Mobile Alliance™, OMA-AD-LightweightM2M-V1_0, URL:http://www.openmobilealliance.org/
[OBSERVE]	Hartke, K. “Observing Resources in CoAP”, IETF RFC 7641.
[PKCS#15]	“PKCS #15 v1.1: Cryptographic Token Information Syntax Standard”, RSA Laboratories, June 6, 2000. URL:ftp://ftp.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
[RFC2234]	“Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt
[RFC4122]	“A Universally Unique Identifier (UUID) URN Namespace”, P. Leach, et al. July 2005, URL:http://www.ietf.org/rfc/rfc4122.txt
[RFC5246]	The Transport Layer Security (TLS) Protocol Version 1.2
[RFC5280]	D. Cooper, et al., “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC 5280, May 2008.
[RFC5289]	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)
[RFC5487]	Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode
[RFC5958]	S. Turner, “Asymmetric Key Packages”, RFC 5958, August 2010.
[RFC6347]	Rescorla, E. and N. Modadugu, “Datagram Transport Layer Security Version 1.2”, RFC 6347 , January 2012.
[RFC6655]	McGrew, D. and D. Bailey, “AES-CCM Cipher Suites for TLS”, RFC6655, July 2012.

[RFC6690]	Shelby, Z. “Constrained RESTful Environments (CoRE) Link Format”, RFC6690, Aug 2012.
[RFC7049]	C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR)”, RFC 7049, October 2013.
[RFC7292]	K. Moriarty, et al., “PKCS #12: Personal Information Exchange Syntax v1.1”, RFC 7292, July 2014.
[SENML]	C. Jennings, Z. Shelby, J. Arkko, “Media Types for Sensor Markup Language (SENML)”, draft-jennings-senml-10 (work in progress), April 2013.
[TR-069]	Broadband Forum: “TR-069 CPE WAN Management Protocol” Issue: 1 Amendment 5. URL: http://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf
[WAP-WDP]	Wireless Application Protocol Forum, “Wireless Datagram Protocol”, June 2001.
[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 (Universal) Subscriber Identity Module (U)SIM Toolkit applications (Release 10)”
[3GPP2 C.S0078-0]	3GPP2 C.S0078-0 (V1.0): “Secured packet structure for CDMA Card Application Toolkit (CCAT) applications”
[3GPP2 C.S0079-0]	3GPP2 C.S0079-0 (V1.0) “Remote APDU Structure for CDMA Card Application Toolkit (CCAT) applications”

[DMREPPRO]	“OMA Device Management Representation Protocol, Version 1.3”. Open Mobile Alliance™. OMA-TS-DM_RepPro-V1_3. URL:http://www.openmobilealliance.org
[DYNAMIC LINK]	“Dynamic Resource Linking for Constrained RESTful Environments”, Z.Shelby, Z.Vial, M.Koster, C.Groves, Oct 2016, draft-ietf-core-dynlink-01
[ETSI TS 102 226]	ETSI TS 102 226 (V11.0.0): “Smart cards; Remote APDU structure for UICC based applications (Release 11)”
[ISO/IEC18031:2011]	ISO, “ISO/IEC 18031:2011: Information technology -- Security techniques -- Random bit generation”, November 2011, available at URL:http://www.iso.org/iso/catalogue_detail.htm?csnumber=54945
[OMADICT]	“Dictionary for OMA Specifications”, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/
[OMNA]	“OMNA Lightweight M2M (LwM2M) Object & Resource Registry”, URL:http://www.openmobilealliance.org/
[RESOURCE DIRECTORY]	“CoRE Resource Directory”, Z.Shelby, M.Koster, C.Bormann, P.Van Der Stok Oct 2016, draft-ietf-core-resource-directory-09
[RFC3986]	T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, RFC 3986, January 2005.
[RFC4086]	D. Eastlake, J. Schiller, S. Crocker, “Randomness Requirements for Security”, RFC 4086, June 2005.
[RFC6698]	P. Hoffman, J. Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA”, RFC 6698, August 2012.
[RFC7459]	“Representation of Uncertainty and Confidence in the Presence Information Data Format Location Object (PIDF-LO)”, M. Thomson, J. Winterbottom, February 2015. URL:https://tools.ietf.org/html/rfc7459

[SMS-DTLS]	“Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things”, H. Tschofenig, T. Fossati, July 2016, URL:http://www.ietf.org/rfc/rfc7925.txt
[SP800-90A]	Elaine Barker, John Kelsey, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication 800-90A”, Revision 1, June 2015, available at http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

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

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

Kindly consult [\[OMADICT\]](#) for more definitions used in this document.

3.3. Abbreviations

kB	Kilobyte; one kilobyte is 1000 bytes
----	--------------------------------------

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.

Four interfaces are designed between the three entities, as shown in the architecture in Figure 1:

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

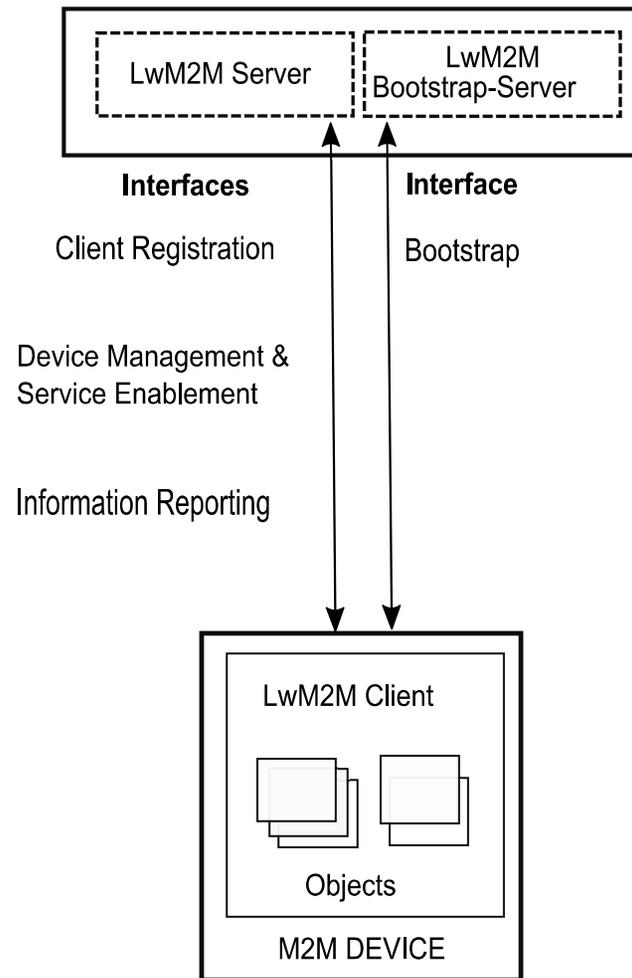


Figure: 4.-1 The overall architecture of the LwM2M Enabler

4.1. Version 1.0

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

- o. Security Object

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

5. Interfaces

According to the architecture diagram [LwM2M-AD], there are four interfaces: 1) Bootstrap, 2) Client Registration, 3) Device Management and Service Enablement, and 4) Information Reporting. The operations for the four interfaces can be classified as uplink operations and downlink operations. The operations of each interface are defined in this section, and then mapped to protocol mechanisms in Section 8 Transport Layer Bindings and Encodings.

Figure 3 shows the operation model for interface “Bootstrap”. For this interface, the operations are an uplink operation named “Bootstrap-Request” and downlink operations named “Discover”, “Write”, “Delete” and “Bootstrap-Finish”. These operations are used to initialize the needed Object(s) for the LwM2M Client to register with one or more LwM2M Servers. With the “Write” operation on this interface, the LwM2M Client MUST write the value included in the payload regardless of an existence of the targeting Object Instance(s) or Resource(s) and access rights. In the mode where the Server is addressing the Bootstrap Information to the LwM2M Client, the Server MUST inform the LwM2M Client when this transfer is over by sending a “Bootstrap-Finish” command.

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

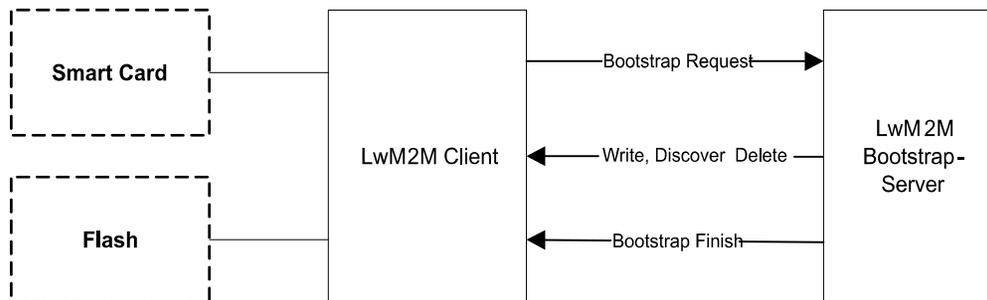


Figure: 5.-1 Bootstrap

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

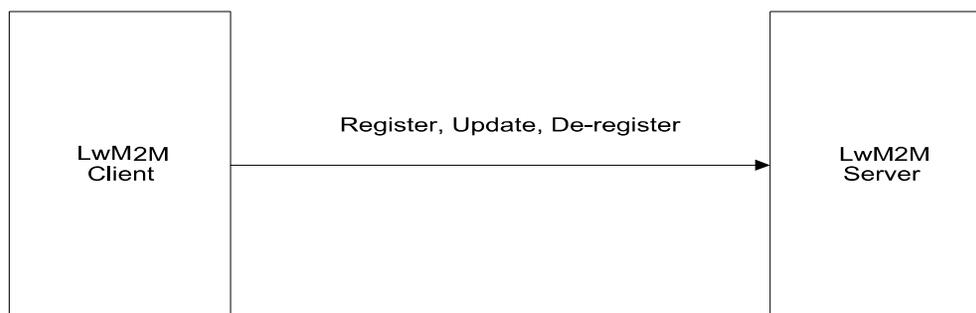


Figure: 5.-2 Client Registration

Figure 5 shows the logical operation model for interface “Device Management and Service Enablement”. For this interface, the operations are downlink operations named “Read”, “Create”, “Delete”, “Write”, “Execute”, “Write-Attributes”, and “Discover”. These operations are used to interact with the Resources, Resource Instances, Objects, Object Instances and/or their attributes exposed by the LwM2M Client. The “Read” operation is used to read the current values; the “Discover” operation is used to discover attributes and to discover which Resources are implemented in a certain Object; the “Write” operation is used to update the values; the “Write-Attributes” operation is used to change attribute values and the “Execute” operation is used to initiate an action. The “Create” and “Delete” operations are used to create or delete Instances.

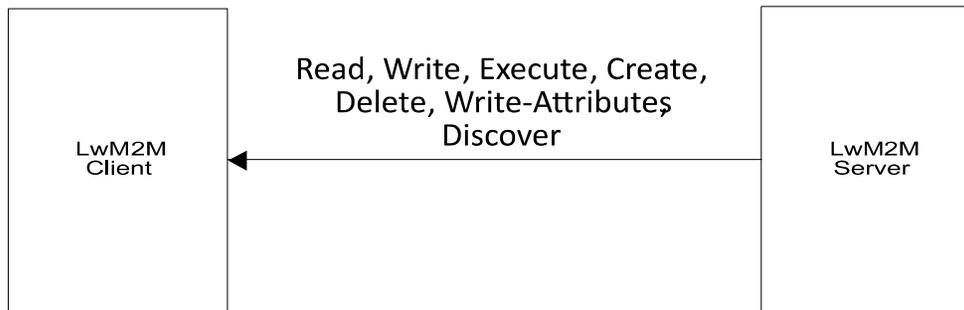


Figure: 5.-3 Device Management and Service Enablement

Figure 6 shows the operation model for interface “Information Reporting”. For this interface, the operations are downlink operations “Observe” or “Cancel Observation” and an uplink operation “Notify”. This interface is used to send the LwM2M Server a new value related to a Resource on the LwM2M Client.

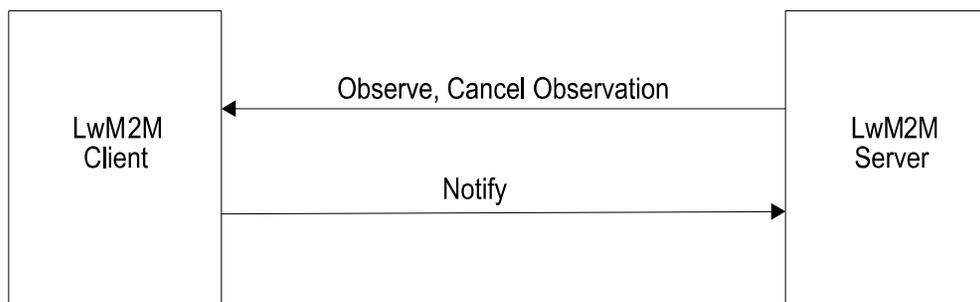


Figure: 5.-4 Information Reporting

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

Interface	Direction	Operation
Bootstrap	Uplink	Bootstrap-Request
Bootstrap	Downlink	Write, Discover, Delete, Bootstrap-Finish

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

Table: 5.-1 Relationship of operations and interfaces

5.1. Attributes

5.1.1. Attributes Definitions and Rules

Attributes are metadata which can be attached to an Object, an Object Instance or a Resource. The value of an Attribute is LwM2M Server specific. These attributes can fulfil various roles: from just carrying information (e.g., Discover), up to containing parameters for Notification for example.

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

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

Regardless to the LwM2M entity a given Attribute is attached to, the value of such an Attribute can be assigned at various levels: Object, Object Instance, Resource levels. Additionally, precedence rules apply when the same Attribute receives a value at different levels.

Several rules govern usage of LwM2M Attributes

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

precedence rules:

- Rule 1: When set at both levels, the value of the OI-Attribute set at Object Instance level will prevail.
- Rule 2: When the Attribute value is set at the Object level, the scope of the OI-Attribute value extends to all the Instances of that Object, as long as the Rule 1 is respected.
- An R-Attribute MAY be set at 4 different levels: the Ressource 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 the table below:

Attribute characteristics	Description
Name	Attribute Name used to reference a specific Attribute in that Enabler (e.g., “Minimum Period”)
CoRE Link Param	the string used when this Attribute is transferred to a CoRE link parameter (e.g., pmin)
Attachment	The Object, Object Instance or Resource, to which an Attribute is logically applied
Assignment 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 TS 1.1 <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)
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” command (Section 5.4.2).

The value of some Attributes MAY be changed by the LwM2M Server in using the “Write-Attributes” command (Section 5.4.4); which Attribute are concerned are marked as “W” (writable) in the table of the next Section.

Note: A payload response to a “Discover” command is a list of application/link-format CoRE Links [RFC6690] which will include the LwM2M Attributes.

5.1.2. Attributes Classification

<PROPERTIES> Class Attributes

The role of these Attributes is to provide metadata which may communicate helpful information to LwM2M Server for example easing data management.

Except when specifically mentioned as required, the LwM2M Server and LwM2M Client SHOULD support <PROPERTIES> Class Attributes listed Table 3.

Attribute Name	CoRE Link param	Attachment	Assignment Level	Support Required	Access Mode	Value Type	Default Value	Applicability	Notes
Dimension	dim	Resource	Resource	YES (Client)	R	Integer [0:255]	-	Multiple-Instance Resource	Number of Instantiations for a Multiple Resource
Object Version	ver	Object	Object	YES (Server) YES (Client) when able to support Objects in version > 1.0	R	String	1.0 (Initial Version)		Provide the version of the associated Object. The rules governing the usage of this parameter are specified in Section 6.2 and Table 19.

Table: 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 Conditions ("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 command, a coherency check of the 2 rules above MUST lead to reject that command 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 LwM2M Server MUST support and LwM2M Client SHOULD support all the <NOTIFICATION> Class Attributes listed Table 4.

Attribute Name	CoRE Link param	Attachment	Assignment Level	Required	Access Mode	Value Type	Default Value	Apply Condition
Minimum Period	pmin	Resource	Resource Instance Resource Object Instance Object	No	RW	Integer	0 (sec)	Readable Resource
<p><i>Notes:</i> The Minimum Period Attribute indicates the minimum time in seconds the LwM2M Client MUST wait between two notifications. If a Resource value has to be notified during the specified quiet period, the notification MUST be sent as soon as this period expires. In the absence of this parameter, the Minimum Period is defined by the Default Minimum Period set in the LwM2M Server Account.</p>								
Maximum Period	pmax	Resource	Resource Instance Resource Object Instance Object	No	RW	Integer	-	Readable Resource
<p><i>Notes:</i> 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 maximum period parameter MUST not be smaller than the minimum period parameter otherwise it will be ignored for the Resource to which such inconsistent Timing Conditions applied.</p>								
Greater Than	gt	Resource	Resource Instance Resource	No	RW	Float	-	Numerical & Readable Resource
<p><i>Notes:</i> 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).</p>								

Less Than	lt	Resource	Resource Instance Resource	No	RW	Float	-	Numerical & Readable Resource
<p><i>Notes:</i> 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).</p>								
Step	st	Resource	Resource Instance Resource	No	RW	Float	-	Numerical & Readable Resource
<p><i>Notes:</i> 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).</p>								
Minimum Evaluation Period	epmin	Resource	Resource Instance Resource Object Instance Object	No	RW	Integer	-	Readable Resource
<p><i>Notes:</i> 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.</p>								
Maximum Evaluation Period	epmax	Resource	Resource Instance Resource Object Instance Object	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 8.2.5 (Device Management & Service Enablement Interface).

5.2. Bootstrap Interface

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

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

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

The last two Bootstrap modes require the help of a LwM2M Bootstrap-Server to achieve the ultimate goal to connect a LwM2M Client to their LwM2M Server(s).As specified in section 5.2.3.4, 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.

5.2.1. LwM2M Bootstrap-Server

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

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

5.2.2. Bootstrap Information

This section specifies the information that needs to be configured in LwM2M Client for connecting to the LwM2M Server(s) or the LwM2M Bootstrap-Server. This Bootstrap Information can be available before performing the Bootstrap Sequence described in Section 5.2.4 or obtained as a result of the Bootstrap Sequence.

Bootstrap Information can be categorized into two types:

- LwM2M Server Bootstrap Information
- LwM2M Bootstrap-Server Bootstrap Information

The LwM2M Client MUST have the LwM2M Server Bootstrap Information after the Bootstrap Sequence specified in Section 5.2.4.

The LwM2M Client SHOULD have the LwM2M Bootstrap-Server Bootstrap Information.

The LwM2M Server Bootstrap Information is used by the LwM2M Client to register and connect to the LwM2M Server.

The LwM2M Server Bootstrap Information MUST contain at least a LwM2M Server Account.

Note that according to the LwM2M Server Account definition, a usual LwM2M Server Account is composed of a Security Object Instance and a Server Object Instance which are paired by sharing (respectively in Resource 10 and Resource 0 of

that Objects) the same Short Server ID; a Short Server ID being unique in the LwM2M Client.

The LwM2M Server Bootstrap Information MAY additionally contain further Object Instances (e.g., Access Control, Connectivity Monitoring Object).

The LwM2M Client MAY be configured to use one or more LwM2M Server Account(s).

The LwM2M Client MUST have at most one LwM2M Bootstrap-Server Account.

The LwM2M Bootstrap-Server Bootstrap Information is used by the LwM2M Client to contact the LwM2M Bootstrap-Server to get the LwM2M Server Bootstrap Information.

The LwM2M Bootstrap-Server Bootstrap Information MUST be a LwM2M Bootstrap-Server Account.

Bootstrap Information Type	Entity	Required
The LwM2M Server Bootstrap Information	LwM2M Server Account	Yes*
	Additional Object Instances (e.g., Access Control, Connectivity Monitoring Object)	No
The LwM2M Bootstrap-Server Bootstrap Information	LwM2M Bootstrap-Server Account (Security Object instance)	No

Table: 5.2.2.-1 Bootstrap Information List

*the LwM2M Client MUST have at least one LwM2M Server Account after completion of Bootstrap Sequence specified in 5.2.4.

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

5.2.3. Bootstrap Modes

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

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

5.2.3.1. Factory Bootstrap

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

5.2.3.2. Bootstrap from Smartcard

When the Device supports a Smartcard, the LwM2M Client MUST retrieve and process the bootstrap data contained in the Smartcard as described in Appendix G. When the bootstrap data retrieval is successful, the LwM2M Client MUST process the bootstrap data from the Smartcard and SHOULD apply the Bootstrap Information to its configuration to enhance security benefits.

Due to the sensible nature of the Bootstrap Information, a secure channel SHOULD be established between the Smartcard and the LwM2M Device.

When such a secure channel is established between the Smartcard and the LwM2M Device, this secure channel MUST be based on [GLOBALPLATFORM] procedure, mainly described in Appendix H.

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

If the Smartcard is disabled (e.g., removing the Smartcard) then the Bootstrap Information created from the bootstrap data of the previous Smartcard MUST be deleted.

Checking for Smartcard change and disabling MUST be performed by LwM2M Client, each time a “Register” or “Update” operation take place, with a LwM2M Server provisioned from Smartcard. As usual, the Bootstrap security rules (5.2.5) then apply.

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

5.2.3.3. Client Initiated Bootstrap

As defined in Section 5.2.4 Bootstrap Sequence, scenarios exist when the LwM2M Server is not configured within the LwM2M Client or attempts to perform the “Register” operation with LwM2M Servers have failed.

When these conditions occur, the "Client Initiated Bootstrap" mode provides a mechanism for the LwM2M Client to retrieve the Bootstrap Information from a LwM2M Bootstrap-Server.

The "Client Initiated Bootstrap" mode requires a LwM2M Bootstrap-Server Account preloaded in the LwM2M Client.

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

The figure below depicts the "Client Initiated Bootstrap" flow.

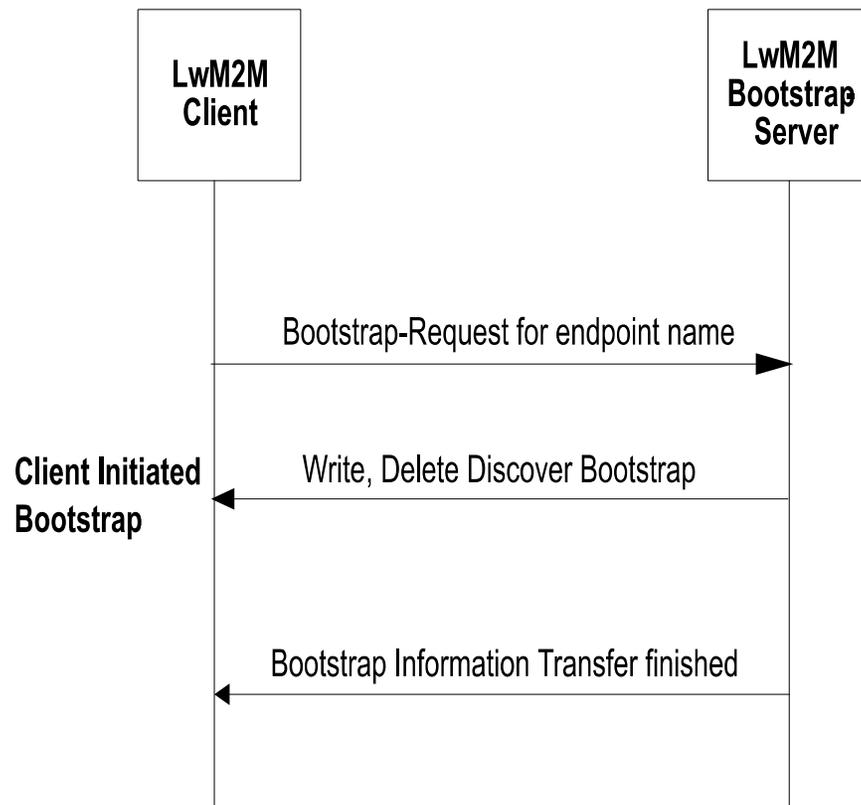


Figure: 5.2.3.3.-1 Procedure of Client Initiated Bootstrap

Step #0: Bootstrap-Request to bootstrap URI

The LwM2M Client sends a “Bootstrap-Request” operation to LwM2M Bootstrap-Server URI which has been pre-provisioned. When requesting the bootstrap, the LwM2M Client sends the LwM2M Client’s “Endpoint Client Name” as a parameter to allow the LwM2M Bootstrap-Server to provision the proper Bootstrap Information for the LwM2M Client.

Step #1: Configure Bootstrap Information

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

This Bootstrap mode MAY be used to configure some Resources of the Bootstrap Information in the LwM2M Client after initial bootstrap to update Bootstrap Information. In this case, all the Bootstrap Information is OPTIONAL.

Step #2: Bootstrap-Finish Indication

When the LwM2M Server has finished to send all the Bootstrap Information to the LwM2M Client, the Server MUST send a Finish Bootstrap Indication to the Client to properly end this phase.

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

EXCHANGE_LIFETIME is defined in RFC 7252 [CoAP].

Step #3: Clean-up after successful Bootstrapping

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

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

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

In case the Bootstrap-Server Account has to be replaced, the replacement and the purge of the previous Bootstrap-Server Account MUST properly take place before the Client sends the Bootstrap-Finish response message back to the Bootstrap-Server; otherwise a "4.06 Not Acceptable" Response code MUST be returned, and the previous Bootstrap-Server Account is still the only one active.

Note: If the original LwM2M Bootstrap-Server Account is purged from the device, and a new LwM2M Bootstrap-Server Account has NOT been created, further adding or removing of LwM2M Server Accounts will no longer be possible. Furthermore, updating security credentials e.g., X.509 certificates will also no longer be possible.

5.2.3.4. Server Initiated Bootstrap

In this mode, the decision to trigger a Bootstrap Sequence is made by an authorized LwM2M Server. However, 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 by an authorized LwM2M Server through the execution of the "Bootstrap-Request Trigger" Resource available from the related Server Object Instance; this can be considered a "secondary" Bootstrap mode, since 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 push 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.

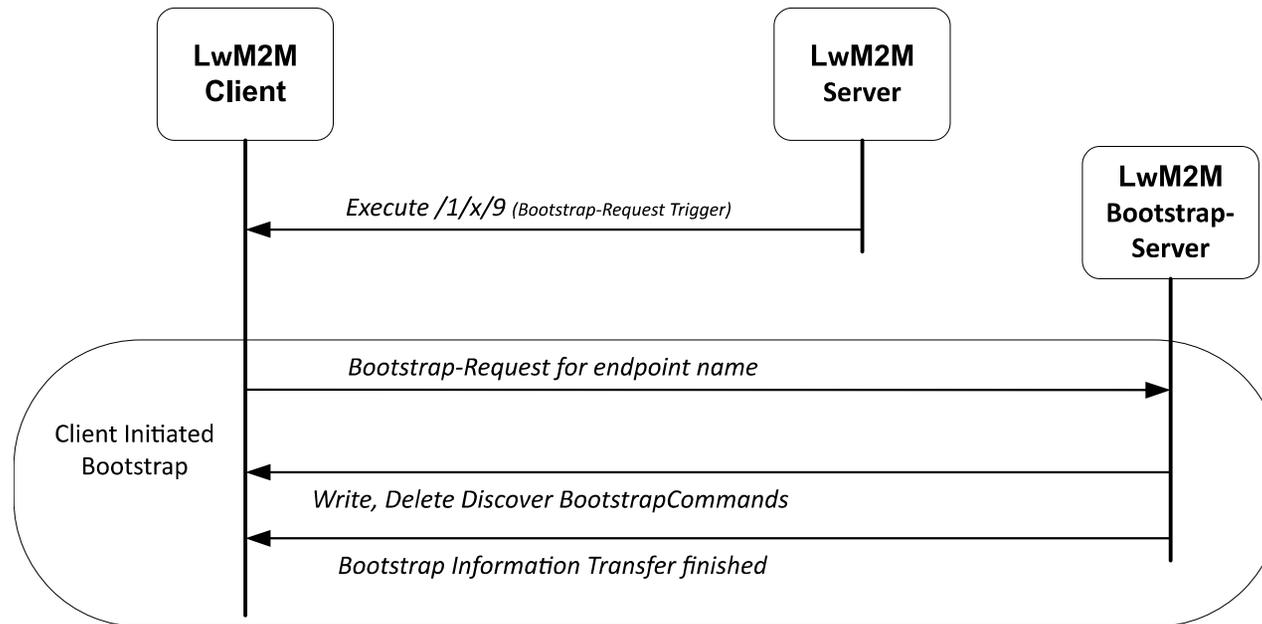


Figure: 5.2.3.4.-1 Procedure of Server Initiated Bootstrap initiated by an authorized LwM2M Server

5.2.4. Bootstrap Sequence

The LwM2M Client MUST respect step by step the procedural sequence specified below when attempting to bootstrap a LwM2M Device:

1. If the LwM2M Device has Smartcard, the LwM2M Client tries to obtain Bootstrap Information from the Smartcard using the Bootstrap from Smartcard mode. Any Server Initiated Bootstrap attempt MUST be ignored by the LwM2M Client until it has tried to bootstrap via Smartcard or Factory Bootstrap mode.
2. If the LwM2M Client is not configured using the Bootstrap from Smartcard mode, the LwM2M Client tries to obtain the Bootstrap Information by using Factory Bootstrap mode. Any Server Initiated Bootstrap attempt MUST be ignored by the LwM2M Client until it has tried to bootstrap via Smartcard or Factory Bootstrap mode.
3. If the LwM2M Client has any LwM2M Server Object Instances from the previous steps, the LwM2M Client tries to register to the LwM2M Server(s) configured in the LwM2M Server Object Instance(s).
4. If LwM2M Client fails to register to all the LwM2M Servers or the Client doesn't have any LwM2M Server Object Instances, the LwM2M Client performs the Client Initiated Bootstrap.
5. A Server Initiated Bootstrap attempt (e.g., for updating an LwM2M Server Account) remains possible, but only if the LwM2M Client retains the corresponding LwM2M Bootstrap-Server Account.

5.2.5. Bootstrap Security

The information conveyed through the Bootstrap Interface is sensitive and requires that communication session, security mechanisms and/or keys MUST be different instances from the one that is used for the other LwM2M Interfaces.

If the LwM2M Client or the LwM2M Bootstrap-Server needs to convey Bootstrap Information across the Bootstrap Interface, the LwM2M Client or the LwM2M Bootstrap-Server MUST establish a new secure communication session.

If security materials (e.g., LwM2M Server URI, Security Mode, and Security Key), are changed in the LwM2M Client, the LwM2M Client MUST disconnect the existing communication session between the LwM2M Server and LwM2M Client and establish a new secure communication session between the LwM2M Server and LwM2M Client using the security mechanism and/or keys which have been configured by Bootstrap Interface.

5.2.6. Bootstrap and Configuration Consistency

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

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

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

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

5.2.7. Bootstrap Commands

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

The Bootstrap Commands 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).

5.2.7.1. *BOOTSTRAP-REQUEST*

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

The Bootstrap-Request operation has the following parameter:

Parameter	Required	Default Value	Notes
/bs?ep={Endpoint Client Name}	Yes	-	Indicates the LwM2M Client’s “Endpoint Name” in order to allow the LwM2M Bootstrap to provision the Bootstrap Information for the LwM2M Client.

Table: 5.2.7.1.-1 Bootstrap-Request operation parameter

5.2.7.2. *BOOTSTRAP-FINISH*

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

The Bootstrap-Finish operation has the following parameter:

Parameter	Required	Default Value	Notes
/bs	Yes	-	-

Table: 5.2.7.2.-1 Bootstrap-Finish operation parameter

5.2.7.3. *BOOTSTRAP DISCOVER*

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

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

The returned payload is a list of application/link-format CoRE Links [RFC6690] containing the LwM2M Enabler Version and for each targeted Object – in addition to the Object Version (see section 6.2 “Object Versioning” and Table 3 of this document <PROPERTIES> 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 doesn’t 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 "Discover" command the targeted path '/' is accepted in place of the Object ID, for informing the Client to report all existing Object Instances.

The “Discover” operation has the following parameters:

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

Table: 5.2.7.3.-1 Bootstrap Discover parameters

For example:

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

`lwm2m="1.0",</0/0>;ssid=101;uri="coaps://server_1.example.com", </0/1/>, </0/2>;ssid=102;uri="coaps://server_2.example.com" with the meaning 3 LwM2M Servers are supported in that Client (that supports the LwM2M Enabler in version 1.0), while the Instance ID:1 of the Security Object ID:0 contains the credentials for the LwM2M Bootstrap-Server.`

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

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

- When the “Discover” addresses a LwM2M Client supporting the LwM2M Enabler in release 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>`

5.2.7.4. 5.2.7.4 BOOTSTRAP READ

The “Read” operation in Bootstrap Interface is a restricted form of the "Read" operation in Device Management and Service Enablement interface, and MUST be limited to target Objects which are strictly necessary to setup a proper configuration in a LwM2M Client; typically in LwM2M 1.1, the only acceptable targets for the Bootstrap "Read" 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 Rights already in place in the targeted LwM2M Client.

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

The “Read” operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object. In LwM2M 1.1, 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 read. If no Object Instance ID is indicated, then all Instances of the targeted Object are returned

As a simple illustration based on the second example of section 6.4.3.2 (Multiple Object Instance Request Examples): the Bootstrap "Read /2" command with JSON requested format could provide the following answer:

```
{“bn”：“/2“,
“e”:\[
{"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}
]
}
```

5.2.7.5. BOOTSTRAP WRITE

The “Write” operation in Bootstrap Interface is different from the “Write” Operation in Device Management and Service Enablement interface; the LwM2M Client MUST write the value included in the payload regardless of an existence of the targeted Object Instance(s) or Resource(s).

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

The Write operation can be sent multiple times.

Only in Bootstrap Interface, the “Write” MAY target just an Object ID, which will allow a Bootstrap-Server in using a TLV, CBOR or JSON formatted payload, to populate a LwM2M Client in a single message containing several Instances of the same Object.

The Bootstrap “Write” operation has the following parameters:

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

Table: 5.2.7.5.-1 Bootstrap “Write” operation parameters

5.2.7.6. *BOOTSTRAP DELETE*

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

Only in Bootstrap Interface, the 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 and the single Instance of the Mandatory Device Object (ID:3) which are not affected by any Delete operation.

When the 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 purpose before LwM2M Bootstrap-Server sends Write operation(s) to the LwM2M Client.

The Delete operation has the following parameters:

Parameter	Required	Default Value	Notes
Object ID	No	-	Indicates the Object from which Object Instance will be deleted; if no Object ID is indicated, all existing Object Instances (except the LwM2M Bootstrap-Server Account 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: 5.2.7.6.-1 Delete operation parameters

5.3. Client Registration Interface

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

The Client Registration Interface (Note) is used by a LwM2M Client to register with one or more LwM2M Servers, maintain each registration and de-register from a LwM2M Server. The registration is based on the Resource Model and Identifiers defined in Section 6 Identifiers and Resources. When registering, the LwM2M Client performs the “Register” operation and provides the properties the LwM2M Server requires to contact the LwM2M Client (e.g., End Point Name); maintain the registration and session (e.g., Lifetime, Queue Mode) between the LwM2M Client and LwM2M Server as well as knowledge of the Objects the LwM2M Client supports and existing Object Instances in the LwM2M Client. The registration is soft state, with a lifetime indicated by the Lifetime Resource of that LwM2M Server Object Instance. The LwM2M Client periodically performs an update of its registration information to the registered LwM2M Server(s) by performing the “Update” operation—possibly without any parameters. If the lifetime of a registration expires without receiving an update from the LwM2M Client:

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

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

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

Finally, when shutting down or discontinuing use of a LwM2M Server, the LwM2M Client performs a “De-register” operation.

The Binding Resource of the LwM2M Server Object informs the LwM2M Client of the transport protocol preferences of the LwM2M Server for the communication session between the LwM2M Client and LwM2M Server. The LwM2M Client SHOULD perform the operations with the modes indicated by the Binding Resource of the LwM2M Server Object Instance.

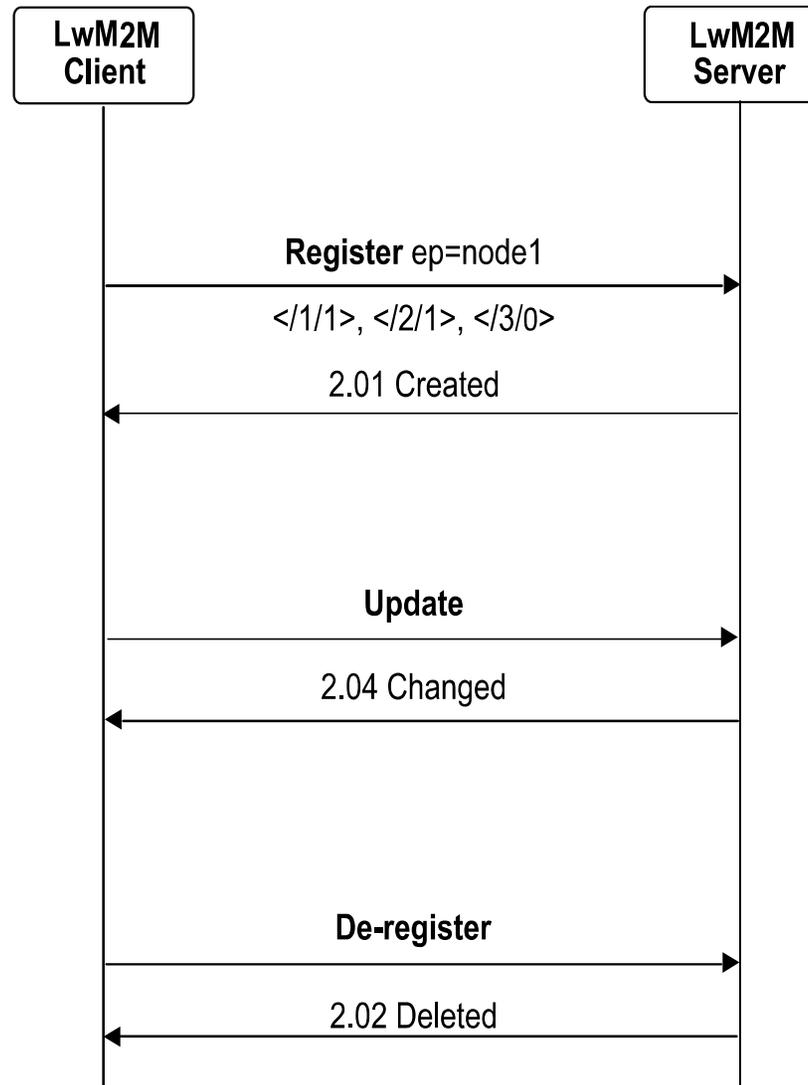


Figure: 5.3.-1 Client Registration Interface example flows

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

5.3.1. Register

Registration is performed when a LwM2M Client sends a “Register” operation to the LwM2M Server. After the LwM2M Device is turned on and the bootstrap procedure has been completed, the LwM2M Client MUST perform a “Register” operation to each LwM2M Server that the LwM2M Client has a Server Object Instance. Table 7 describes the parameters used for the “Register” operation.

The “Register” operation includes the Endpoint Client Name parameter along with other parameters listed in Table 7. The “Register” operation MUST include a value for the Endpoint Client Name parameter that is unique on that LwM2M Server.

Upon receiving a “Register” operation from the LwM2M Client, the LwM2M Server records the connection information of the registration message (e.g., source IP address and port or MSISDN) and uses this information for all future interactions with that LwM2M Client.

If the LwM2M Client sends a “Register” operation to the LwM2M Server even though the LwM2M Server has registration information of the LwM2M Client, the LwM2M Server removes the existing registration information and performs the new “Register” operation. This situation happens when the LwM2M Client forgets the state of the LwM2M Server (e.g., factory reset).

The LwM2M Server MUST support all the parameters listed at Table 7 and the LwM2M Client MUST support Endpoint Client Name, LwM2M Version, Lifetime, and Object and Object Instances and MAY support Binding Mode and SMS Number.

Parameter	Required	Default Value	Notes
Endpoint Client Name	Yes		See Section 6.3
Lifetime	Yes		Indicates the expected lifetime of the registration for this LwM2M client. This value MUST be the same as the value held in the Resource named “Lifetime” of the corresponding instance of Server Object (ID \#1): /1/x/1.

LwM2M Version	Yes		Indicates the version of the LwM2M Enabler that the LwM2M Client supports.
Binding Mode	No	U	Indicates current binding and Queue mode of the LwM2M Client. This value MUST be the same as the value held in the Resource named “Binding” of the corresponding instance of Server Object (ID \#1): /1/x/7. The valid values of the parameter are listed in the Section 5.3.1.1.
SMS Number	No		The value of this parameter is the MSISDN where the LwM2M Client can be reached for use with the SMS binding.
Objects and Object Instances	Yes		The list of Objects supported and Object Instances available on the LwM2M Client (Security Object ID:0 MUST not be part of this list).

Table: 5.3.1.-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 (i.e. major digit of the LwM2M versions in Client and Server are different, or the LwM2M version supported by the Client – e.g., 1.1 – is superior to the LwM2M version supported by the Server – e.g., 1.0 –).

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

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

The payload Media-Type of that registration message MUST be the Core Link Format (application/link-format) defined in [RFC6690], so that each Object is described as a Link according to that format. The Target component of the link is

required, and consists of the Object path augmented of the Object Version Core link parameters “ver” if required as it is defined in section 6.2 of this document. Any other parameters included in the link MUST be silently ignored, unless specified for use by the LwM2M Enabler.

The payload for a LwM2M Client supporting LwM2M Server, Access Control, Device, Connectivity Monitoring and Firmware Update Objects from Appendix E would simply be:

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

If Objects Instances are already available on the LwM2M Client at the time of registration, then the format would be (for the example client of Appendix F):

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

If the LwM2M Client supports the JSON data format for all the Objects it SHOULD inform the LwM2M Server by including the content type in the root path link using the ct= link attribute. An example is as follows (note that the content type value 11543 is the value assigned in the CoAP Content-Format Registry for the LwM2M JSON format).

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

5.3.1.1. Behaviour with Current Transport Binding and Mode

Behaviour of the LwM2M Server and the LwM2M Client is differentiated by Current Transport Binding and Mode. Current Transport Binding and Mode is decided by “Binding” Resource set by the LwM2M Server and whether SMS and/or Queue Mode are supported by the LwM2M Client. Queue Mode is useful when the LwM2M Device is not reachable by the LwM2M Server at all the times and it could help the LwM2M Client sleep longer. Table 8 describes the behaviour of the LwM2M Server and the LwM2M Client for each Current Transport Binding and Mode.

Current Transport Binding and Mode	Behaviour
------------------------------------	-----------

U (UDP)	<p>The LwM2M Server expects that the LwM2M Client is reachable via the UDP binding at any time.</p> <p>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.</p> <p>This is the normal default mode of operation.</p>
UQ (UDP with Queue Mode)	<p>The Server MUST queue all requests to the LwM2M Client, sending requests via UDP when the LwM2M Client is on-line as described in Section 8.4 Queue Mode Operation.</p> <p>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.</p>
S (SMS)	<p>The LwM2M Server expects that the LwM2M Client is reachable via the SMS binding at any time.</p> <p>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.</p>
SQ (SMS with Queue Mode)	<p>The Server MUST queue all requests to the LwM2M Client, sending requests via SMS when the LwM2M Client is on-line as described in Section 8.4 Queue Mode Operation.</p> <p>Requests MUST be sent to the LwM2M Client using the SMS binding. The LwM2M Client MUST send the response to such a request over the SMS binding.</p>
US (UDP and SMS)	<p>The LwM2M Server expects that the LwM2M Client is reachable via the UDP binding at any time.</p> <p>The LwM2M Server expects that the LwM2M Client is reachable via the SMS binding at any time.</p> <p>If the LwM2M Server sends requests to a LwM2M Client using the UDP binding, The LwM2M Client MUST send the response to such a request over the UDP binding.</p> <p>If the LwM2M Server sends requests to a LwM2M Client using the SMS binding, The LwM2M Client MUST send the immediate response to such a request over the SMS binding.</p>

<p>UQS (UDP with Queue Mode and SMS)</p>	<p>The Server MUST queue all requests to the LwM2M Client, sending requests via UDP when the LwM2M Client is on-line as described in Section 8.4 Queue Mode Operation.</p> <p>The LwM2M Server expects that the LwM2M Client is reachable via the SMS binding at any time.</p> <p>If the LwM2M Server sends requests to a LwM2M Client using the UDP binding, The LwM2M Client MUST send the response to such a request over the UDP binding.</p> <p>If the LwM2M Server sends requests to a LwM2M Client using the SMS binding, The LwM2M Client MUST send the immediate response to such a request over the SMS binding.</p> <p>The LwM2M Server MAY request the LwM2M Client to perform “Update” operation via UDP by sending “Execute” operation on “Registration Update Trigger” Resource via SMS.</p>
<p>UQT (UDP with Queue Mode and SMS Trigger)</p>	<p>The Server MUST queue all requests to the LwM2M Client, sending requests via UDP when the LwM2M Client is on-line as described in Section 8.4 Queue Mode Operation.</p> <p>The LwM2M Server expects that the LwM2M Client is reachable via the SMS binding at any time (only for the purposes of sending an SMS Trigger).</p> <p>If the LwM2M Server sends requests to a LwM2M Client using the UDP binding, The LwM2M Client MUST send the response to such a request over the UDP binding.</p> <p>The LwM2M Server MAY request the LwM2M Client to perform “Update” operation via UDP by sending “Execute” operation on “Registration Update Trigger” Resource via SMS. No SMS response to this request is expected.</p>

Table: 5.3.1.1.-1 Behaviour with Current Transport Binding and Mode

UQSQ and USQ are not supported.

5.3.2. Update

Periodically or based on certain events within the LwM2M Client or initiated by the LwM2M Server, the LwM2M Client updates its registration information with a LwM2M Server by sending an “Update” operation to the LwM2M Server.

The “Update” operation can be initiated by the LwM2M Server via an “Execute” operation on the “Registration Update Trigger” Resource of the LwM2M Server Object. The LwM2M Client can perform an “Update” operation to refresh the lifetime of its registration to a LwM2M Server.

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

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

Table: 5.3.2.-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 MUST NOT be part of Update Objects and Object Instances list.

The payload Media-Type of that “Update” message is the same as the one of the “Registration” message (section 5.3.1).

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

Two common operations are:

1. Extending the lifetime of a registration

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

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

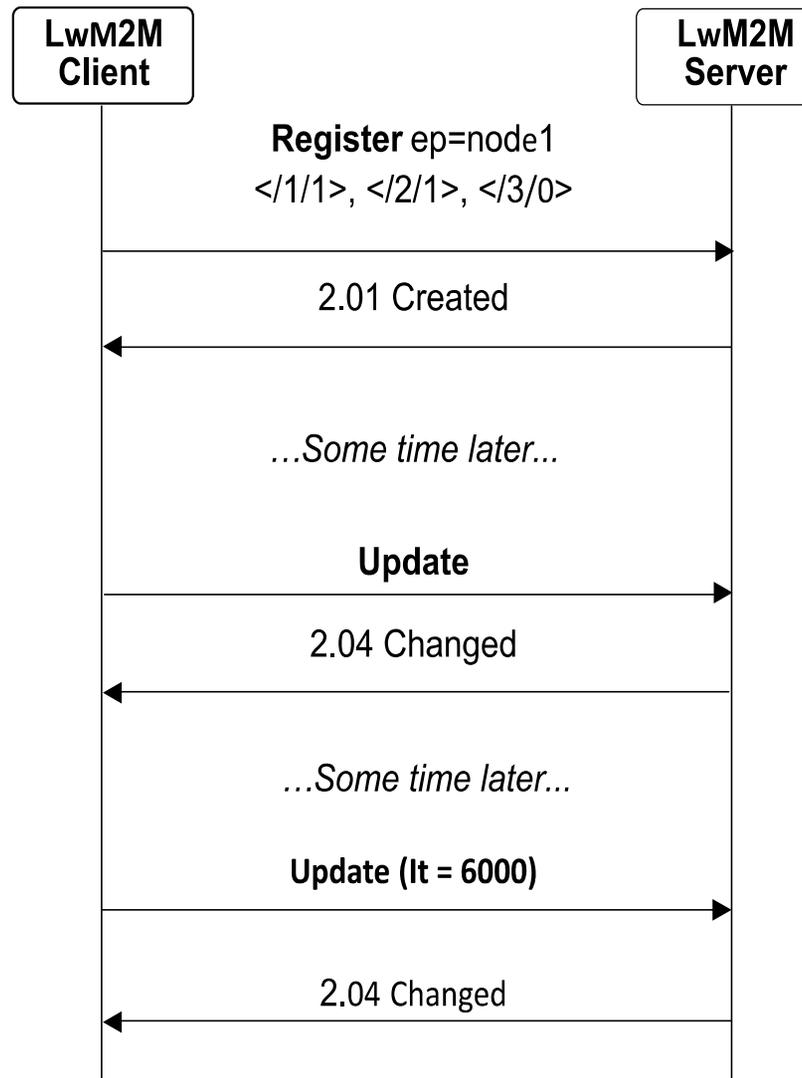


Figure: 5.3.2.-1 Client Registration Update example flows #1

2. Adding and removing Objects and Object Instances

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

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

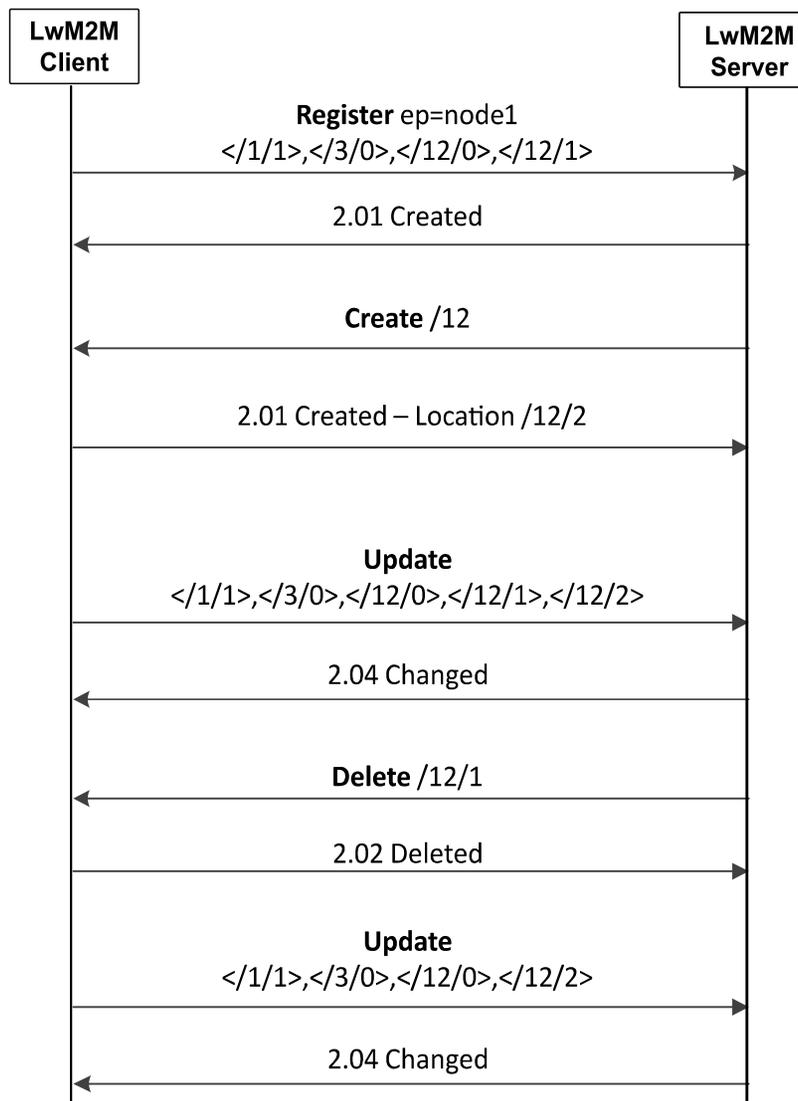


Figure: 5.3.2.-2 Client Registration Update example flows #2

5.3.3. De-register

When a LwM2M Client determines that it no longer requires to be available to a LwM2M Server (e.g., LwM2M Device factory reset), the LwM2M Client SHOULD send a “De-register” operation to the LwM2M Server. Upon receiving this message, the LwM2M Server removes the registration information from the LwM2M Server.

5.4. Device Management & Service Enablement Interface

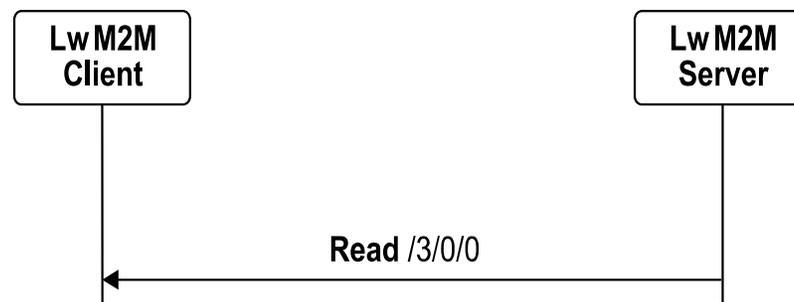
The LwM2M Server and the LwM2M Client MUST support all the operations on this interface unless clearly stated otherwise. Support for some of the non-mandatory operations may also be dependent on the choice of transport layer and the services it provides.

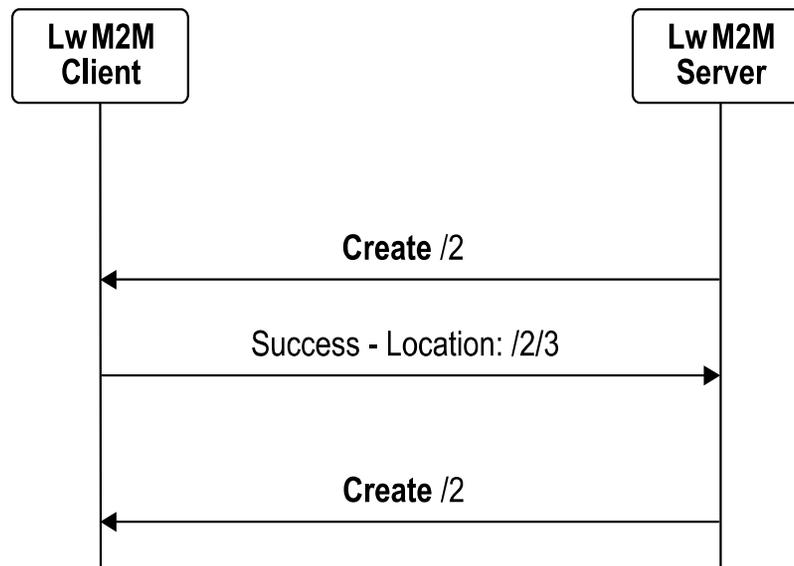
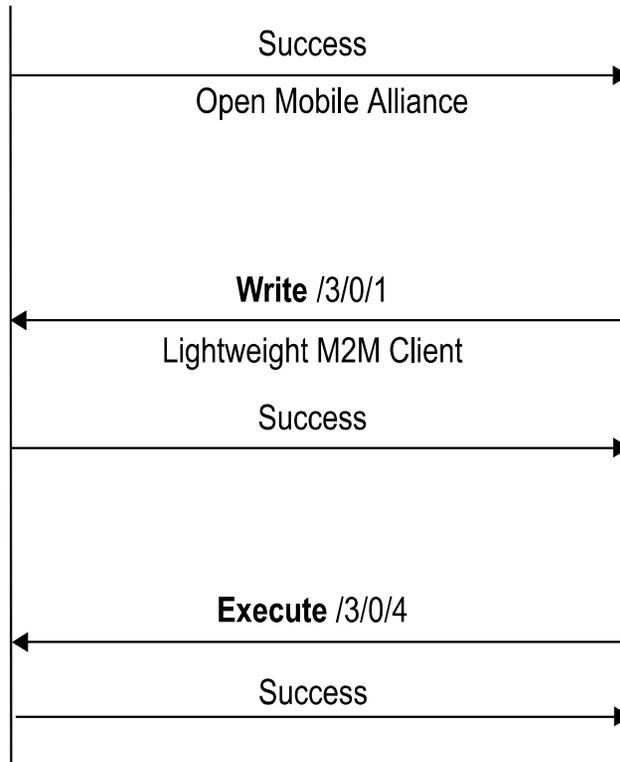
The Device Management and Service Enable Interface (Note) is used by the LwM2M Server to access Object Instances and Resources available from a registered LwM2M Client. The interface provides this access through the use of “Create”, “Read”, “Write”, “Delete”, “Execute”, “Write-Attributes”, or “Discover” operations. The operations that Resource supports are defined in the Object definition using the Object Template. The Object Template is described in Appendix D.1 Object Template. The Normative Objects defined by the LwM2M Enabler are described in Appendix E.

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

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

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





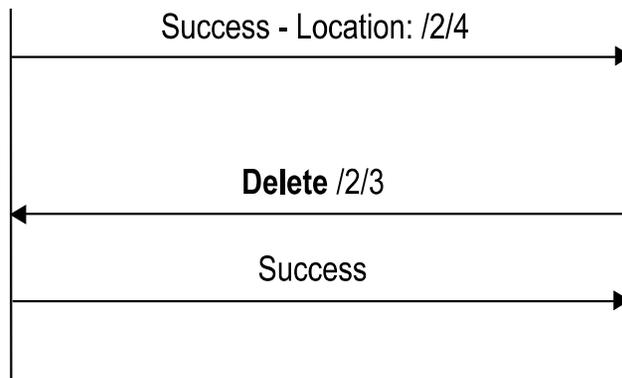


Figure: 5.4.-1 Example flows of Device Management & Service Enablement Ineterface

Note : The mapping of CoAP Methods of the LwM2M Client Registration Interface operations specified in this section, is detailed in Transport TS (Transport Layer Binding and Encodings)

5.4.1. Read

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 Ressource ID and Ressource 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: 5.4.1.-1 Read parameters

5.4.2. Discover

The “Discover” operation is used to discover LwM2M Attributes attached to an Object, Object Instances, and Resources. This operation can be used to discover which Resources are instantiated in a given Object Instance. The returned payload is a list of application/link-format CoRE Links [RFC6690] for each targeted Object, Object Instance, or Resource, along with their assigned or attached Attributes including the Object Version attribute if required (see section 6.2 “Object Versioning” and Table 3 of this document <PROPERTIES> Class Attributes).

The “Discover” operation has the following parameters:

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

Table: 5.4.2.-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 (see section 5.1.2 Attribute Classification) are also returned.

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

</3>;pmin=10,</3/0>,</3/0/1>,</3/0/2>,</3/0/3>,</3/0/4>,</3/0/6>,</3/0/7>,</3/0/8>,</3/0/11>,</3/0/16> which means that the LwM2M Client supports the Device Info Object (Instance 0) Resources with IDs 1,2,3,4 6,7,8,11, and 16 among the Resources of Device Info Object, with an R-Attributes assigned to the Object level.

- 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, 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 request. As example, for the Device Object (ID:3) having the Resource ID:7 with two Observe operations from two Servers 1 & 2, then the answers to DISCOVER command on both Servers will be differentiated e.g.,

from Server 1: DISCOVER /3/0/7 could provide the answer: </3/0/7>;dim=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>

5.4.3. Write

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 6.4.

The TLV, CBOR or 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.

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 deleted to the condition the LwM2M Client authorizes such operations

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: 5.4.3.-1 Write parameters

5.4.4. Write-Attributes

In LwM2M 1.x, only Attributes from the <NOTIFICATION> class MAY be changed in using the “Write-Attributes” operation.

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

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

Including <NOTIFICATION> class Attributes specified in Table 3 Section 5.3.1, the “Write-Attributes” operation has the following parameters:

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

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance. When this parameter is omitted, the Resource ID and Resource Instance ID parameters MUST NOT be indicated and Attributes in the parameters of the command are valid for all resources of all instances of the Object according to the precedence rules (section 5.1.1).

Resource ID	No	-	Indicates the Resource. When this parameter is omitted, then the ResourceInstance ID MUST NOT be indicated as well and it means the Attributes of this commands 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 command (Object, Object Instance, Resource, or Resource Instance levels).

Table: 5.4.4.-1 Write-Attributes parameters

5.4.5. Execute

The “Execute” operation is used by the LwM2M Server to initiate some action, and can only be performed on individual Resources. A LwM2M Client MUST return an error when the “Execute” operation is received for an Object Instance(s) or Resource Instance(s).

Resource which supports “Execute” operation MAY have arguments.

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

Resource ID	Yes	-	Indicates the Resource to execute.
Arguments	No	-	The arguments of the “Execute” operation are expressed in Plain Text format (syntax below).

Table: 5.4.5.-1 Execute parameters

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

```

arglist = arg* ("," arg)

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

```

Figure: 5.4.5.-1 ABNF syntax

Examples of valid lists of arguments

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

5.4.6. Create

The “Create” operation is used by the LwM2M Server to create Object Instance(s) within the LwM2M Client. The “Create” operation MUST target an Object, and MUST follow the rules specified in section 7.3 (ACCESS CONTROL) and its sub-sections. If any error occurs, nothing MUST be created.

The Object Instance created in the LwM2M Client by the LwM2M Server MUST be an Object type supported by the LwM2M Client and announced to the LwM2M Server using the “Register” and “Update” operations of the LwM2M Client Registration Interface.

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

The “Create” operation has the following parameters:

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

Table: 5.4.6.-1 Create parameters

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

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

When there is no reference to Object Instance in the TLV/CBOR/JSON payload of the “Create” command, the LwM2M Client MUST assign the ID of the created Object Instance. If a new Object Instance is created through that operation and the LwM2M Client has more than one LwM2M Server Account, then the LwM2M Client creates an Access Control Object Instance for the created Object Instance (7.3 ACCESS CONTROL)

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

5.4.7. Delete

The “Delete” operation is used for LwM2M Server to delete an Object Instance within the LwM2M Client.

The Object Instance that is deleted in the LwM2M Client by the LwM2M Server MUST be an Object Instance that is announced by the LwM2M Client to the LwM2M Server using the “Register” and “Update” operations of the Client Registration Interface.

The 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: 5.4.7.-1 Delete parameters

5.4.8. Read-Composite

If supported by the client, the “Read-Composite” operation can be used by the LwM2M Server to read a number of resources, and/or resource instances of different objects in a single request. The list of elements to be read are provided as separate parameters to the operation in JSON format. 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 will not

be included in the response. The following represents the BNF description of the parameters for "Read-Composite", see also the section on [JSON](#) for actual examples.

```
Read_Composite_Parameters =addressed_element * [ "," addressed_element]
```

```
addressed_element = <obj_id>"\"<instance_id>[\"\"<resource_id>[\"\"<resource_instance_id]]
```

5.4.9. Write-Composite

In contrast to "write" operation the scope of which is limited to a single object and one or more of its instances, or a single resource, 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. The list of all resources to be updated are provided as name value pairs in JSON format; see the section on [JSON](#) for actual examples. The "Write-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.

5.5. Information Reporting Interface

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

The Information Reporting Interface (Note) is used by a LwM2M Server to observe any changes in a Resource on a registered LwM2M Client, receiving notifications when new values are available. This observation relationship is initiated by sending an "Observe" operation to the LwM2M Client for an Object, an Object Instance or a Resource. An observation ends when a "Cancel Observation" operation is performed.

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

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

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

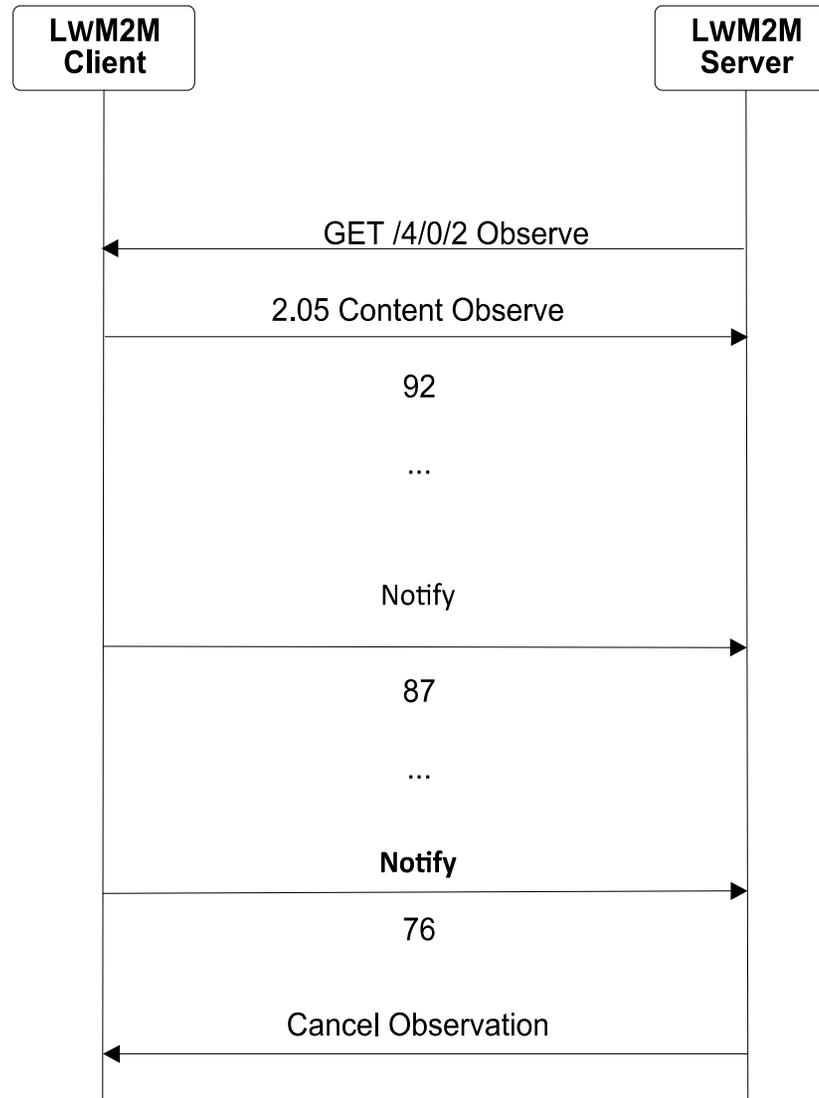


Figure: 5.5.-1 Example flow for Information Reporting Interface for the RSSI Resource of the Connectivity Monitoring Object of the example client (Appendix E)

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

5.5.1. Observe

The LwM2M Server initiates an observation request for changes of a specific Resource, Resources within an Object Instance or for all the Object Instances of an Object within the LwM2M Client.

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

The Observe operation includes the following parameters:

Parameter	Required	Default Value	Notes
Object ID	Yes	-	Indicates the Object.
Object Instance ID	No	-	Indicates the Object Instance to observe. If no Object Instance ID is indicated, then 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: 5.5.1.-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.

5.5.2. Notify

The “Notify” operation is sent from the LwM2M Client to the LwM2M Server during a valid observation on an Object Instance or Resource. This operation includes the new value of the Object Instance or Resource. The “Notify” operation SHOULD be sent when all the conditions (i.e., Minimum Period, Maximum Period, Greater Than, Less Than, Step) configured by “Write-Attributes” operation for “Observe” operation are met.

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

Table: 5.5.2.-1 Notify parameters

The following example shows how the Minimum and Maximum period parameters work as shown in Section 5.4.4. A LwM2M Server makes an observation for a Temperature Resource that is updated inside the LwM2M Client at irregular periods (based on change). The LwM2M Server makes an observation when the Minimum Period = 10 Seconds and Maximum Period = 60 Seconds have been set for that Resource. The LwM2M Client will wait at least 10 Seconds before sending a “Notify” operation to the LwM2M Server (even if the Resource has changed before that), and no longer than

60 Seconds before sending a “Notify” operation (even if the Resource has not changed yet). The “Notify” operation is sent anywhere between 10–60 seconds upon change.

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 5.5.1 and in Table 17, in case of an “Observe” operation on a certain Object the “Notify” operation includes all Resources of all Instances of the observed Object.

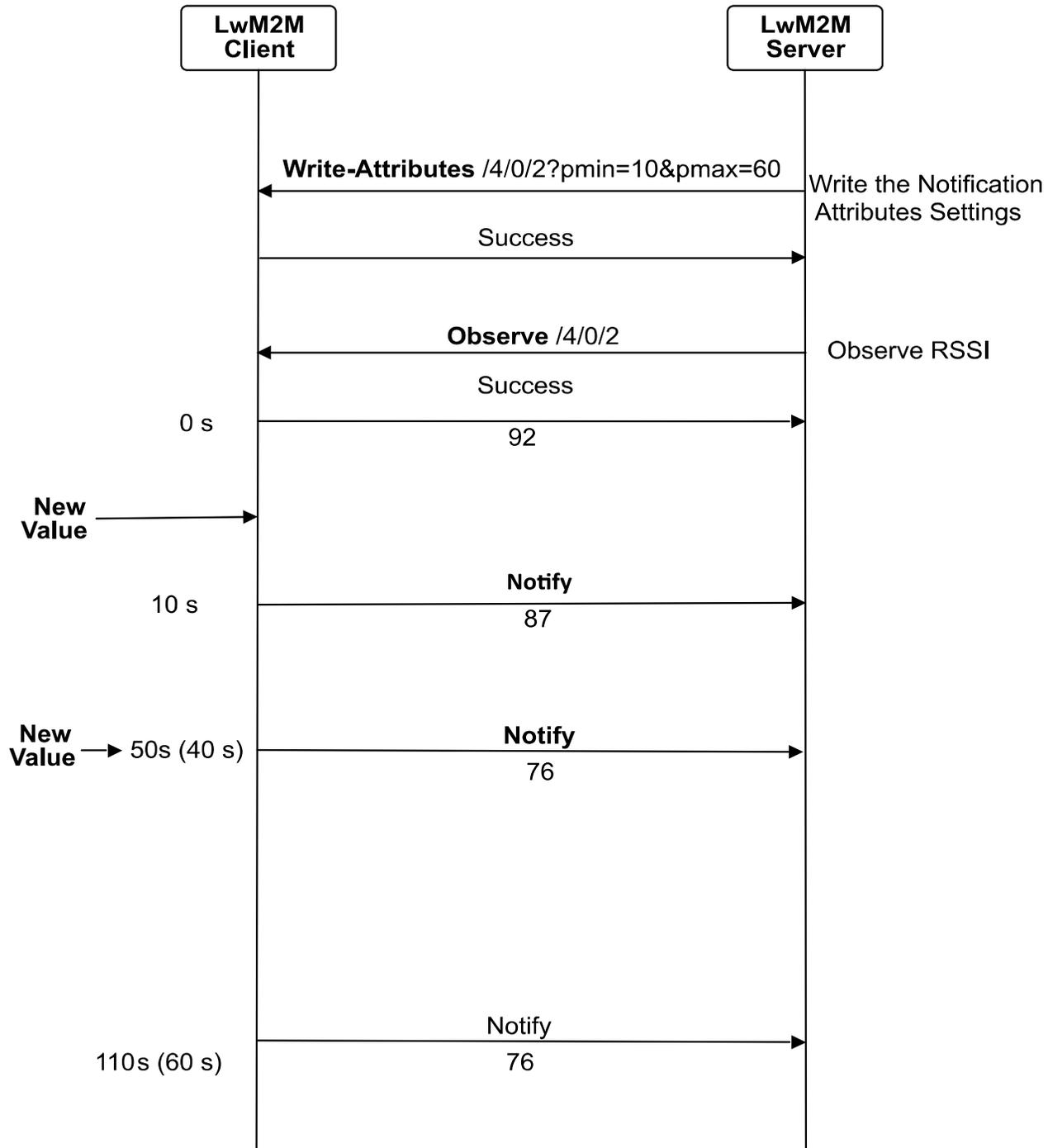


Figure: 5.5.2.-1 Example of Minimum and Maximum periods in an Observation

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

5.5.3. Cancel Observation

The “Cancel Observation” operation is sent from the LwM2M Server to the LwM2M Client to end an observation relationship for Object Instance or Resource.

5.5.4. Observe-Composite

If supported by the client, the LwM2M Server can use “Observe-Composite” operation to initiate observation of a group of resources and/or resource instances across multiple object instances within the client. As in "Read-Composite" operation the list of elements to be observed are provided as separate parameters to the operation in the same Jason format. The following represents the BNF description of the parameters.

```
Observe_Composite_Parameters =addressed_element * ["," addressed_element]
```

```
addressed_element = <obj_id>"\"<instance_id>[\"\"<resource_id>[\"\"<resource_instance_id]]
```

Notification class attributes for each resource that is observed through "Observe-Composite" are individually configured by the server using "Write-Attribute" and as before 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 client which will include the value for each of the resources listed in "Observe-Composite". That is the resources that have not met notify condition will still be included in the notification message. "Observe-Composite" can be cancelled in the same way as a normal "Observe"

6. Identifiers and Resources

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

6.1. Resource Model

The LwM2M Enabler defines a simple resource model where each piece of information made available by the LwM2M Client is a Resource. Resources are logically organized into Objects. Each Resource is given a unique identifier within that Object and a data type among those defined in Appendix C (Data Types).

Figure 15 illustrates this structure, and the relationship between Resources, Objects, and the LwM2M Client. The LwM2M Client may have any number of Resources, each of which belongs to an Object; for example the Firmware Update Object contains all the Resources used for firmware update purposes.

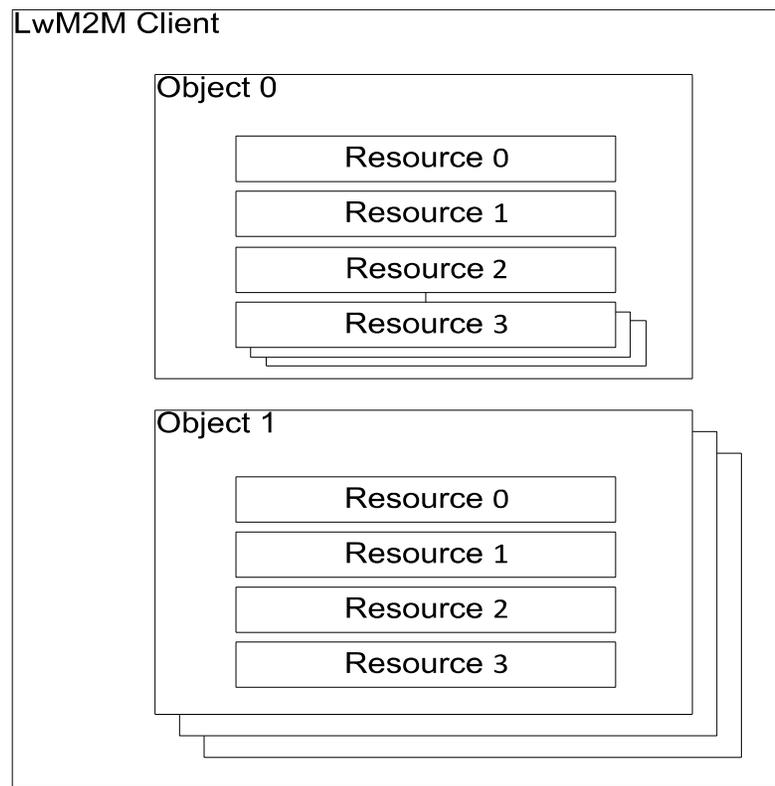


Figure: 6.1. -1 Relationship between LwM2M Client, Object, and Resources

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

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

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

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

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

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

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

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

Objects and Resources have the capability to have multiple instances. Multiple-Instance Resources can be instantiated by LwM2M Server operations in using JSON, CBOR or TLV formats (Section 5.4). 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 5 Interfaces. These operations are conveyed as described in Section 8 Transport Layer Binding and Encoding and how to convey the Operation data is defined in 6.4.

The LwM2M Enabler defines an access control mechanism per Object Instance. Object Instances SHOULD have an associated Access Control Object Instance. An Access Control Object Instance contains Access Control Lists (ACLs) that define which operations on a given Object Instance are allowed for which LwM2M Server(s).

Figure 16 shows an example of the operations the Resources support and how Object Instances and Resources are associated with Access Control Object Instance. In the example, Object Instance 0 for Object 1 has 2 Resources. Resource 1 supports the "Read", "Write" operations, while Resource 0 supports only the "Read" operation. The associated Access Control Object Instance has ACL of Object Instance 0 for Object 1. Server1 is authorized to perform "Read" and "Write" operations to the Object Instance 0 for Object 1 and Resources of the Object Instance. However, due to the supported operations of each Resource, Server1 can perform the "Read" operation on Resource 1 and 0, and also can perform the "Write" operations on Resource 1, but Server1 cannot perform the "Write" operation on Resource 0. The detailed access control mechanism is defined in Section 7.3 Access Control.

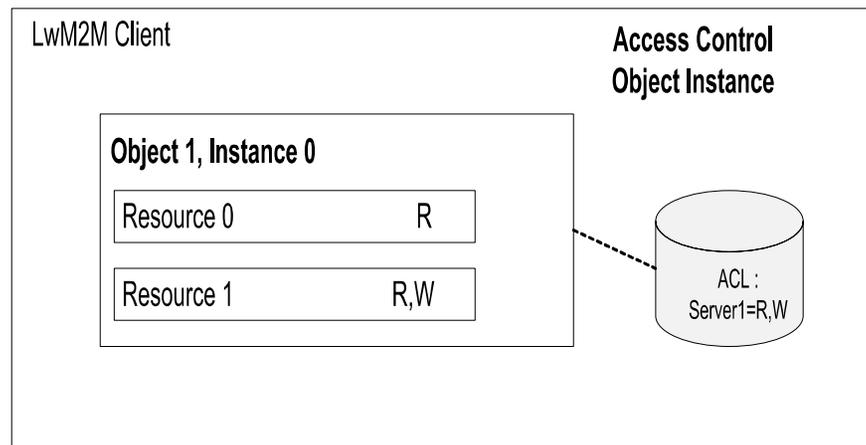


Figure: 6.1.-2 Example of Supported operations and Associated Access Control Object Instance

6.2. Object Versioning

6.2.1. General Policy

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

For example:

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

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

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

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

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

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

In this document, the term “Initial Version” represents:

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

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

“urn:oma:lwm2m:{oma, ext, x}:ObjectID[:{version}]” where ‘version’ is the Object Version as defined in the following section

6.2.2. Object Version format

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

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

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

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

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

Example: “urn:oma:lwm2m:oma:44:2.2”

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

6.2.3. Object Definition and Object Version Usage

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

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

Table: 6.2.3.-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, LwM2M Server ignores minor releases)
- the Object definition contains the minimal version of the LwM2M Enabler supporting that Object (which can be omitted if this LwM2M version is the “Initial Version” one)
- LwM2M Client “Register” operation: ep=nodename,</1/0>,</1/1>,</3/0>,</44>;ver= “2.2”,</44/0>

b) URN: “urn:oma:lwm2m:oma:3”

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

c) URN: “urn:oma:lwm2m:oma:44”

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

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

Details:

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

e) URN: “urn:oma:lwm2m:oma:4:3.1”

Context:

- the LwM2M Client supports LwM2M Enabler version 2.0

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

Details:

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

6.3. Identifiers

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

Identifier	Semantics	Description
Endpoint Client Name	URN	Identifies the LwM2M Client on one LwM2M Server (including LwM2M Bootstrap-Server). Provided to the LwM2M Server during Registration, also provided to LwM2M Bootstrap-Server when executing the Bootstrap procedure. Recommended URN formats are documented in Section 6.3.1 Endpoint Client Name.
LwM2M Bootstrap-Server URI	URI	Uniquely identifies the LwM2M Bootstrap-Server. Provided to the LwM2M Client during the Bootstrap procedure.
LwM2M Server URI	URI	Uniquely identifies the LwM2M Server. Provided to the Client during Bootstrap procedure.

Short Server ID	16-bit unsigned integer	Uniquely identifies each LwM2M Server configured for the LwM2M Client. The identifier is assigned during the Bootstrap procedure. The values '0' and MAX_ID '65535' are reserved values and MUST NOT be used for identifying the LwM2M Server.
Human Readable Object URN	URN for the OMA Management Object	Assigned by the Object specification.
Object ID	16-bit unsigned integer	Uniquely identifies the Object in the LwM2M Client. This identifier is assigned by OMA. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the Object.
Object Instance ID	16-bit unsigned integer	Uniquely identifies the Object Instance of the Object within the LwM2M Client. This identifier is assigned by LwM2M Client or LwM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the Object Instance.
Resource ID	16-bit unsigned integer	Uniquely identifies the Resource within the Object. Short integer ID, with a range assigned by the Object specification and unique to that Object, and a Reusable Resource ID range assigned by OMA and re-usable between Objects. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the Resource.
Resource Instance ID	16-bit unsigned integer	Uniquely identifies the Resource Instance in the Resource. This identifier is assigned by LwM2M Client or LwM2M Server. MAX_ID 65535 is a reserved value and MUST NOT be used for identifying the Resource Instance.

Table: 6.3.-1 LwM2M Identifiers

6.3.1. Endpoint Client Name

Following formats are RECOMMENDED for this identifier to guarantee uniqueness:

Format
<p>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:#####-####-####-####-#####</p>
<p>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>.</p>
<p>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>.</p>
<p>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:#####</p>
<p>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:#####</p>
<p>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:#####</p>
<p>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:#####-#####</p>

Table: 6.3.1.-1 Endpoint Client Name

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

6.3.2. Reusable Resources

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

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

The definition of 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: If the operators/vendors want to extend the resources in a non-standard way, it is advisable to refer OMNA and use the private resource range. This action of using private range, would ensure backward compatibility in case the resource(s) is getting extended in future releases of LwM2M TS, by OMA DM WG.

6.4. Data Formats for Transferring Resource Information

Four data formats are defined by the LwM2M Enabler in this section: plain text, opaque, TLV, CBOR and JSON.

The LwM2M Server MUST support all data formats. The LwM2M Client MUST support the TLV data format. In addition a LwM2M Client MAY choose to support other data formats in the table below i.e. CBOR, JSON, plain text and opaque.

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

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

The IANA registered Media Type supported in LwM2M TS 1.0 are listed in the table below

Data Format	IANA Media Type	Numeric Content-Formats [CoAP]
Plain Text	text/plain	0
Core Link Param	application/link-format	40
Opaque	application/octet-stream	42
TLV	application/vnd.oma.lwm2m+tlv	11542
JSON	application/vnd.oma.lwm2m+json	11543
CBOR	application/vnd.oma.lwm2m2+cbor	TBD

Table: 6.4.-1 IANA registered Media Type supported in LwM2M TS 1.0

6.4.1. Plain Text

The plain text format is used for “Read” and “Write” operations on singular Resources where the value of the Resource is represented as described in Appendix C.

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

```
Req: Get /3/0/0  
  
Res: 2.05 Content  
Open Mobile Alliance
```

This data format has a Media Type of text/plain

6.4.2. Opaque

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

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

6.4.3. TLV

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

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

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

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

Identifier	8-bit or 16-bit unsigned integer as indicated by the Type field.	The Object Instance, Resource, or Resource Instance ID as indicated by the Type field.
Length	0-24-bit unsigned integer as indicated by the Type field.	The Length of the following field in bytes.
Value	Sequence of bytes of Length	Value of the tag. The format of the value depends on the Resource's data type (See Appendix C).

Table: 6.4.3.-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.

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

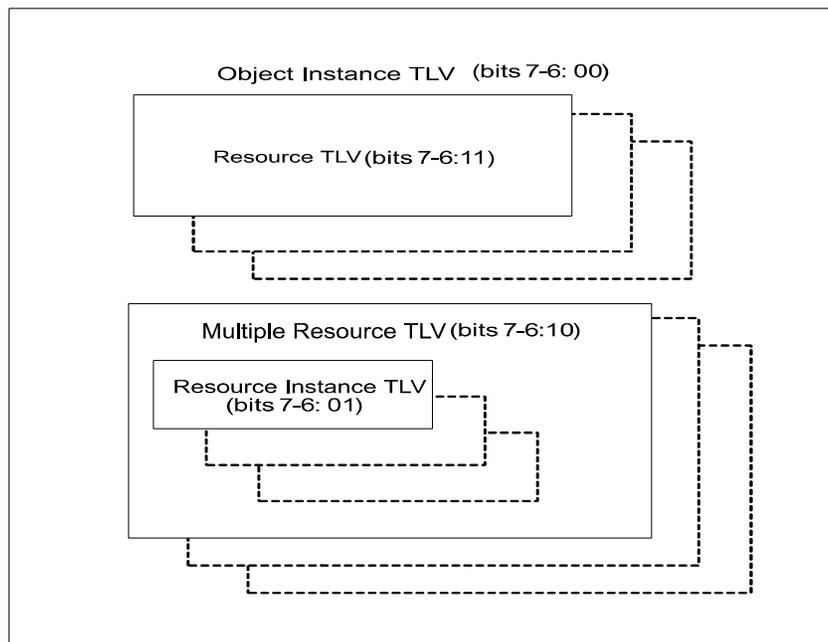


Figure: 6.4.3.-1 TLV nesting

6.4.3.1. Single Object Instance Request Example

In this example, a request for the Device Object Instance (ID:3) of a LwM2M client is made (Read /3/0). The client responds with a TLV payload including all of the readable Resources. This TLV payload would have the following format. The total payload size with the TLV encoding is 121 bytes:

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

```
C8 01 16 4C 69 67 68 74 77 65 69 67 74 20 4D 32 4D 20 43 6C 69 65 6E 74
```

```
C8 02 09 33 34 35 30 30 30 31 32 33
```

```
C3 03 31 2E 30
```

```
86 06
```

41 00 01

41 01 05

88 07 08

42 00 0E D8

42 01 13 88

87 08

41 00 7D

42 01 03 84

C1 09 64

C1 0A 0F

83 0B

41 00 00

C4 0D 51 82 42 8F

C6 0E 2B 30 32 3A 30 30

C1 10 55

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Manufacturer Resource	0b11 0 01 000	0x00	0x14 (20 bytes)	Open Mobile Alliance [String]	23

Model Number	0b11 0 01 000	0x01	0x16 (22 bytes)	“Lightweight M2M Client” [String]	25
Serial Number	0b11 0 01 000	0x02	0x09 (9 bytes)	“345000123” [String]	12
Firmware Version	0b11 0 00 011	0x03	—	“1.0” [String] (3 bytes)	5
Available Power Sources	0b10 0 00 110	0x06	—	The next two rows (6 bytes)	2
Available Power Sources[o]	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[o]	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[o]	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[o]	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: 6.4.3.1.-1 Single Object Instance Request Example

6.4.3.2. Multiple Object Instance Request Examples

A) Request on Single-Instance Object

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

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

08 00 79

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

C8 01 16 4C 69 67 68 74 77 65 69 67 74 20 4D 32 4D 20 43 6C 69 65 6E 74

C8 02 09 33 34 35 30 30 30 31 32 33

C3 03 31 2E 30

86 06

41 00 01

41 01 05

88 07 08

42 00 0E D8

42 01 13 88

87 08

41 00 7D

42 01 03 84

C1 09 64

C1 0A 0F

83 0B

41 00 00

C4 0D 51 82 42 8F

C6 0E 2B 30 32 3A 30 30

C1 10 55

TLV	Type Byte	ID Byte(s)	Length Byte(s)	Value	Total Bytes
Device Object Instance 0	0b00 0 01 000	0x00	0x79 (121 bytes)	The next 20 rows	3
Manufacturer Resource	0b11 0 01 000	0x00	0x14 (20 bytes)	Open Mobile Alliance [String]	23
Model Number	0b11 0 01 000	0x01	0x16 (22 bytes)	“Lightweight M2M Client” [String]	25
Serial Number	0b11 0 01 000	0x02	0x09 (9 bytes)	“345000123” [String]	12
Firmware Version	0b11 0 00 011	0x03	—	“1.0” [String] (3 bytes)	5
Available Power Sources	0b10 0 00 110	0x06	—	The next two rows (6 bytes)	2
Available Power Sources[0]	0b01 0 00 001	0x00	—	0X01 [8-bit Integer]	3
Available Power Sources[1]	0b01 0 00 001	0x01	—	0X05 [8-bit Integer]	3
Power Source Voltage	0b10 0 01 000	0x07	0x08 (8 bytes)	The next two rows	3

Power Source Voltage[0]	0b01 0 00 010	0x00	—	0x0ED8 [16-bit Integer]	4
Power Source Voltage[1]	0b01 0 00 010	0x01	—	0x1388 [16-bit Integer]	4
Power Source Current	0b10 0 00 111	0x08	—	The next two rows (7 bytes)	2
Power Source Current[0]	0b01 0 00 001	0x00	—	0x7D [8-bit Integer]	3
Power Source Current[1]	0b01 0 00 010	0x01	—	0x0384 [16-bit Integer]	4
Battery Level	0b11 0 00 001	0x09	—	0x64 [8-bit Integer]	3
Memory Free	0b11 0 00 001	0x0A	—	0x0F [8-bit Integer]	3
Error Code	0b10 0 00 011	0x0B	—	The next row (3 bytes)	2
Error Code[0]	0b01 0 00 001	0x00	—	0x00 [8-bit Integer]	3
Current Time	0b11 0 00 100	0x0D	—	0x5182428F [32-bit Integer]	6
UTC Offset	0b11 0 00 110	0x0E	—	“+02:00” [String] (6 bytes)	8
Supported Binding and Modes	0b11 0 00 001	0x10	—	“U” [String] (1 byte)	3

Total					124
-------	--	--	--	--	-----

Table: 6.4.3.2.-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
<i>ACL [127]</i>	<i>0b01 0 00 001</i>	<i>0x7F</i>	—	<i>0b000 00111</i> [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
<i>ACL [127]</i>	<i>0b01 0 00 001</i>	<i>0x7F</i>	—	<i>0b000 00111</i> [8-bit Integer]	3
<i>ACL [310]</i>	<i>0b01 1 00 001</i>	<i>0x0136</i>	—	<i>0b000 00001</i> [8-bit Integer]	4
Access Control Owner	0b11 0 00 001	0x03	—	0x7F [8-bit Integer]	3
Total					38

Table: 6.4.3.2.-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 (o) 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 o	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: 6.4.3.2.-3 Example, a request to the Server Object Instances of a LwM2M client is performed (Read /1)

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

Examples are based on the LwM2M Object Tree illustration of Figure 19. The TLV format doesn't report Object hierarchy.

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

88 00 0C

44 00 00 42 00 00

44 01 00 42 00 01

C8 01 0D 38 36 31 33 38 30 30 37 35 35 35 30 30

C4 02 12 34 56 78

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

Res 2	0b11 0 00 100	0x02	—	0x12345678 [32-bit Integer]	6
Total					37

Table: 6.4.3.3.-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: 6.4.3.3.-2 Example 2) request to Object 66: Read /66: TLV payload will contain 2 Object Instances

6.4.4. JSON

When a LwM2M Client is supporting the JSON data format and such a format is used to transport Object Instance(s), multiple resource and single resource values for both “Read” and “Write” operations, JSON payload MUST use the format defined in this section. Such a format MAY be used for transporting a single value of a Resource

The format MUST comply to [SENML] JSON representation extended for supporting LwM2M Object Link data type and MUST support all attributes defined in Table 22.

According to [SENML] semantics, JSON data format in LwM2M, is composed of optional attributes (Base Time, Base Name) and of a mandatory Resource Array having one or more entries. Each array entry contains several optional or mandatory parameters (Name, Time...).

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

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

The JSON is useful for transporting multiple Resource Instances for example when transporting all Instances of an Object with all Resources, and Resource Instances within a single LwM2M Client response.

In particular, when Base Name is set to the LwM2M Object root (e.g., “/”), the JSON format may support to return a hierarchy of Object Instances when Object Link datatype resources are reported (example given below). The resource

instances tree report is performed in using a Breadth-First traversal strategy (see JSON second example below); a given Object Instance MUST appear at most once in that report. The JSON format also includes optional time fields, which allows for multiple versions of representations to be sent in the same payload. The time fields MUST only be used when sending notifications.

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

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

Historical version of notifications are typically generated when “Notification Storing When Disabled or Offline” resource of LwM2M Server Object is set to true (see Appendix D.2) and when the Device comes on line after having been disabled for a period of time.

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

Attributes	JSON Variable	Mandatory?	Description
Base Name	bn	No	The base name string which is prepended to the Name value of the entry for forming a globally unique identifier for the resource.
Base Time	bt	No	The base time which the Time values are relative to.
Resource Array	e Array Parameters	Yes	The Resource list as JSON value array according to [SENML] with Array parameter extension (Object Link).
	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: <ul style="list-style-type: none"> if Base Name is “/”, the Array entry Name of the Resource is {Object}/{Object Instance}/ {Resource}/{Resource Instance} when Base Name is not present, the Array entry Name is the full URI of the requested Resource Instance

	Time	t	No	The time of the representation relative to the Base Time in seconds for a notification. Required only for historical representations.
	Float Value	v	One value field is mandatory	Value as a JSON float if the Resource data type is Integer, Float, or Time.
	Boolean Value	bv		Value as a JSON Boolean if the Resource data type is boolean.
	ObjectLink Value	ov		Value as a JSON string if the Resource data type is Objlnk Format according to Appendix C (e.g “10:03”)
	String Value	sv		Value as a JSON string for all other Resource data types. If the Resource data type is opaque the string value holds the Base64 encoded representation of the Resource.

Table: 6.4.4.-1 JSON format and description

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

```
{“bn”：“/3/0/”,
“e”:[
{"n":“0”,“sv”：“Open Mobile Alliance”},
{"n”：“1”,“sv”：“Lightweight M2M Client”},
{"n”：“2”,“sv”：“345000123”},
{"n”：“3”,“sv”：“1.0”},
{"n”：“6/0”,“v”：1},
{"n”：“6/1”,“v”：5},
{"n”：“7/0”,“v”：3800},
{"n”：“7/1”,“v”：5000},
{"n”：“8/0”,“v”：125},
{"n”：“8/1”,“v”：900},
{"n”：“9”,“v”：100},
{"n”：“10”,“v”：15},
{"n”：“11/0”,“v”：0},
{"n”：“13”,“v”：1367491215},
{"n”：“14”,“sv”：“+02:00”},
{"n”：“16”,“sv”：“U”}]
}
```

Table: 6.4.4. -2 Return JSON payload from example request to Device Object of the LwM2M example client (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 JSON payload:

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

Table: 6.4.4.-3 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 from Figure 19 (Read /65/0) would return the following JSON payload.

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

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

Table: 6.4.4.-4 JSON payload returned from example request to Object 65 of the LwM2M example from Figure 19 (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 JSON payload.

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

Table: 6.4.4.-5 JSON payload returned from example request to Device Object on Resource 0 of the LwM2M example client (Read /3/0/0)

For example a server Read-Composite request to the client to read manufacturer name and battery level from device object together with registration lifetime from the server object will look like

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

Table: 6.4.4.-6 JSON payload in the server request to read manufacturer name, battery level and registration lifetime

In response to the above Read-Composite request the client will return a JSON payload similar to the following

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

Table: 6.4.4.-7 JSON payload in the client response to server request to read manufacturer name, battery level and registration lifetime

For example a server Write-Composite to switch off 2 light sources, dim a 3rd to 20% and set the thermostat to 18 degrees will have a JSON payload as shown in table below. Lights are all controlled by instances of IPSO Light Control Object (Object ID 3311), while thermostat is controlled by an instance of IPSO object Set Point (Object ID 3308).

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

Table: 6.4.4.-8 JSON payload in the server Write-Composite to switch off /3311/0 and /3311/1, while dimming /3311/2

6.4.5. CBOR

When a LwM2M Client is supporting the CBOR data format and such a format is used to transport Object Instance(s), multiple resource and single resource values for both “Read” and “Write” operations, CBOR payload MUST use the format defined in this section. Such a format MAY be used for transporting a single value of a Resource.

The LwM2M CBOR [RFC7049] representation is equivalent to the JSON representation, with the following changes:

- For JSON Numbers, the CBOR representation can use integers, floating point numbers, or decimal fractions (CBOR Tag 4); however a representation SHOULD be chosen such that when the CBOR value is converted back to an IEEE double precision floating point value, it has exactly the same value as the original Number.
- Characters in the String Value are encoded using a definite length text string (type 3).
- For compactness, the LwM2M CBOR representation uses integers for the labels, as defined in the following table.

Name	Label	CBOR Label
Base Name	bn	-2
Base Time	bt	-3
Name	n	0
Value	v	2
String Value	sv	3
Boolean Value	bv	4
Time	t	6

Object Link	ov	9
Resource Array	e	10
+-----+		

Table : LwM2M CBOR representation: integers for map keys

7. Access Control

As the LwM2M Client MAY support one or more LwM2M Servers, there is a need to determine which operation on a certain Object or Object Instance is authorized for which LwM2M Server: Access Control Object is designed for supporting that capability. In the particular case where a single LwM2M Server Account exists in the LwM2M Client, the Server MUST have full access right on all the Objects and Object Instances in the LwM2M Client, and the Access Control Object MAY be not instantiated.

The section 7.3.1 and its sub-sections specify what MUST be applied in multiple LwM2M Servers environment. For consistency and for reducing the efforts of the LwM2M Client when switching from single to multiple LwM2M Servers environment after deployment, the Access Control Object MAY also be instantiated in a single LwM2M Server context. In the case the Access Control Object is instantiated in a single LwM2M Server context, section 7.3.1 and its sub-sections MUST also be applied that context.

7.1. Access Control Object

7.1.1. Access Control Object overview

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

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

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

7.1.2. Access Control Object Management

7.1.2.1. 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 the table below - 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 5 th LSB: "Create" is only configured

3	Access Control Owner	MAX_ID=65535 (meaning: managed by Bootstrap Interface)
---	----------------------	--

Table: 7.1.2.1.-1 Access Control on Object

7.1.2.2. 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 the table below.

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

In particular:

- when a LwM2M Server creates under authorization an Object Instance (see section 7.3.2 Authorization) in the LwM2M Client, an Access Control Object Instance MUST be created in the LwM2M Client with the Resources values which MUST be set as given in the table just below. The Access Control Owner Resource is configured with the Short Server ID of the LwM2M Server.

Resource 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 (Appendix E.3)
3	Access Control Owner	The Short Server ID of the LwM2M Server owner of the associated Object Instance

Table: 7.1.2.2. -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.

The Figure 18 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 ④ which defines the Access Rights applicable to the Instance 2 of the Object X pointed in ④.

In this Access Control Object Instance ④, 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 ④)
- 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 © 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.

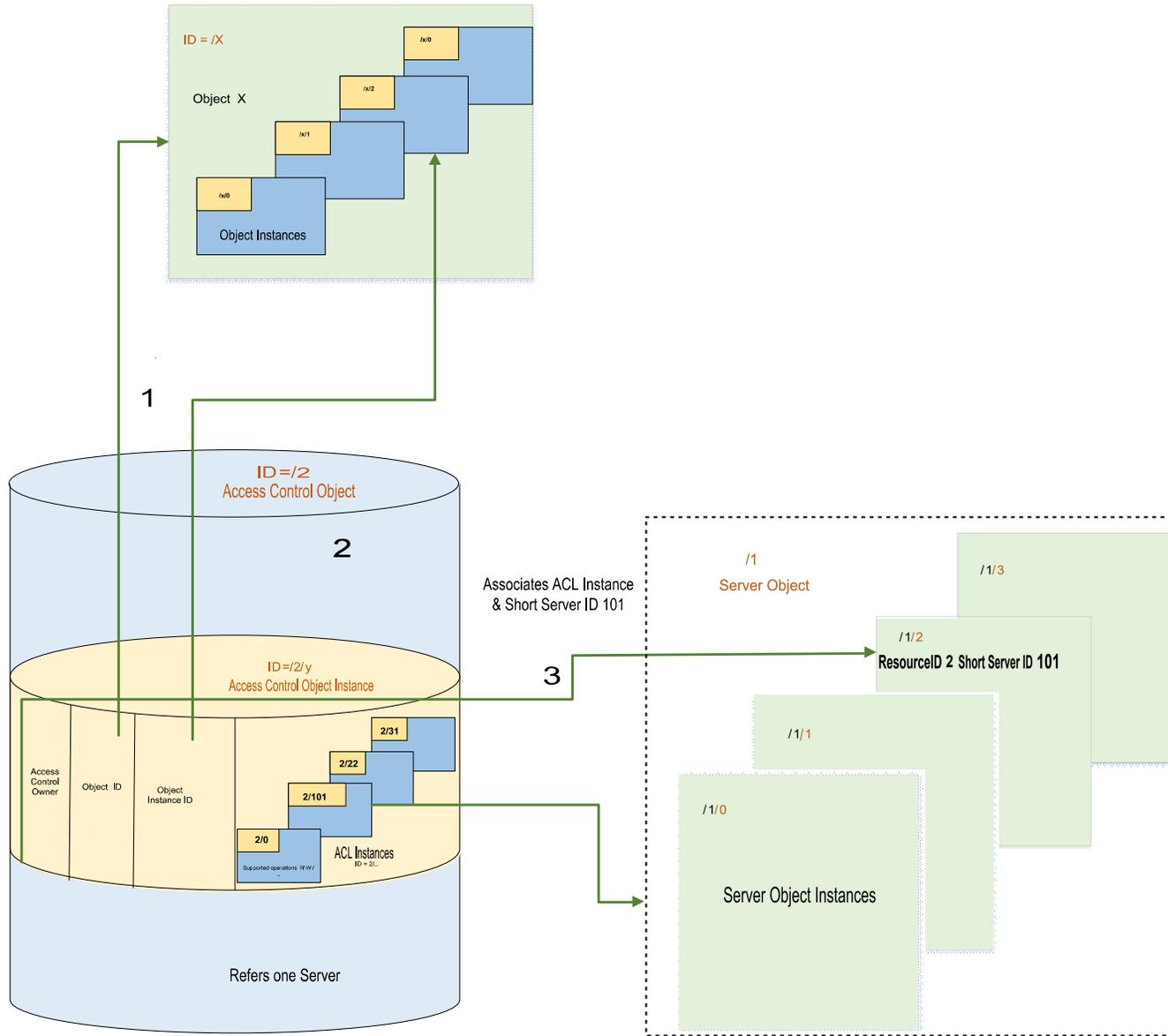


Figure: 7.1.2.2.-1 Illustration of the relations between the LwM2M Access Control Object and the other LwM2M Objects

7.1.3. LwM2M Server Context Switch

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

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

7.2. Authorization

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

The LwM2M Client MUST authorize other operations than CREATE requested by a LwM2M Server either on an Object Instance, or on Resource after performing a two-steps check:

- 1st step: the LwM2M Client gets the access right of the targeted Object Instance (as described in section 7.3.2.1) – and checks whether this access right is sufficient – according to the following table – to perform the LwM2M Server requested operation.

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

Table: 7.2.-1 Authorization

- 2nd step: if at step 1, the LwM2M Server is authorized to perform the operation, the LwM2M Client still needs to check if the LwM2M Server requested operation is supported by the targeted Resource or Object Instance (details are provided in section 7.3.2.2, 7.3.2.3, and 147138816.7.3.27.3.2.4).

The LwM2M Object specification defines which operations are allowed to be performed on Resource within an Object Instance (Refer to Supported Operations in Appendix D LwM2M Object Template and Guidelines). The operations allowed on a given Resource MUST apply to all the Resource Instances of that Resource.

The LwM2M Client MUST support the authorization procedure described in Section 7.3.2 and its sub-sections.

7.2.1. Obtaining Access Right

For “Create” operation sent by the LwM2M Server, the LwM2M Client MUST get access right from the ACL Resource Instance 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 doesn't have the “Create” value, or cannot be obtained, the LwM2M Server has no access right.

For the operations except than “Create” operation the LwM2M Client MUST perform the following procedure:

1. if this LwM2M Server is the only LwM2M Server Account declared in the LwM2M Client (i.e. single Server environment) , the LwM2M Server has full access right on Object Instance(s). When the LwM2M Client gets an Access Control Object Instance associated with the Object Instance the LwM2M Server has requested access to, it MUST proceed according to the sequence below:
 1. If the LwM2M Server is declared as Access Control Owner of this Object Instance and there is no ACL Resource Instance for that Server, then LwM2M Client gets full access right.
 2. If the Client has an ACL Resource Instance for the LwM2M Server, the LwM2M Client gets access right from that ACL Resource Instance.
 3. If the LwM2M Server is not declared as Access Control Owner of this Object Instance and the Client doesn't have ACL Resource Instance for that Server, the LwM2M Client gets access right from the ACL Resource

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

4. If the Client doesn't have this ACL Resource Instance ID:0 containing the default access rights, then the LwM2M Server has no access right, and an "Unauthorized" error code is reported to the LwM2M Server.

7.2.2. Operation on Resource

When the LwM2M Server targets a Resource, the LwM2M Client MUST obtain an access right for the LwM2M Server on the Object Instance that Resource belongs to according to Section 7.3.2.1 and MUST check if the access right is granted prior to perform the requested operation.

- If the operation is not permitted, the LwM2M Client MUST send an "Unauthorized" error code to the LwM2M Server.
- If the operation is permitted, the LwM2M Client verifies if the Resource supports the operation.
 - If the operation is not supported by the Resource, the LwM2M Client MUST send a "Method Not Allowed" error code to the LwM2M Server.
 - If the Resource supports the operation, the LwM2M Client MUST perform the requested operation.

7.2.3. Operation on Object Instance

When the LwM2M Server targets an Object Instance, the LwM2M Client MUST obtain an access right for the LwM2M Server on Object Instance according to Section 7.3.2.1 and MUST check if the access right is granted prior to perform the requested operation.

- If the operation is not permitted, the LwM2M Client MUST send an "Unauthorized" error code to the LwM2M Server.
- If the operation is permitted, the following cases apply, according to the requested operation:
- For the "Write" operation on an Object Instance, the LwM2M Client MUST perform the operation, if all the Resources conveyed in the operation are allowed to perform the "Write" operation. If any Resource does not

support the “Write” operation, the LwM2M Client MUST inform the LwM2M Server that the Object Instance cannot perform the requested “Write” operation in sending a “Operation is not supported” error code.

- For the “Read” and “Observe” operations, the LwM2M Client MUST retrieve all the Resources except the Resource(s) which doesn’t support the “Read” operation and sends the retrieved Resource(s) information to the LwM2M Server.
- For the “Execute” operation, the LwM2M Client MUST NOT perform the operation, and MUST send an “Operation is not supported” error code to the LwM2M Server.
- For the “Delete”, “Write-Attributes”, and “Discover” operations, the LwM2M Client MUST perform the operation.

7.2.4. Operation on Object

If a given LwM2M Server targets an Object with a “Write”, “Execute”, or “Delete” operation, the LwM2M Client MUST NOT perform such an operation and MUST send 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 on Object according to Section 7.3.2.1 “Obtaining Access Right” and MUST check if the access right is granted prior to perform the requested operation.

If the “Create” operation is permitted, the LwM2M Client MUST perform the instantiation on the Object only if all the mandatory Resources are present in the “New Value” parameter (see Section 5). If all the mandatory Resources are not present, the LwM2M Client MUST send a “Bad Request” error code to the LwM2M Server.

Optional Resources MAY be conveyed in the “New Value” parameter as well; the LwM2M Client MAY ignore the optional resources it doesn’t support. The values of the Read-only Resources MUST be setup by the LwM2M Client only; if a value of such a Read-only Resource is present in the “New Value” parameter, this value MUST simply be ignored. If the payload (New Value) conveys an Object Instance ID in conflict with one already present in the LwM2M Client, the complete request MUST be rejected and a “Bad Request” error code MUST be sent back.

- The “Discover” operation on Object is specific in the sense, that no access right is needed; the LwM2M Client MUST perform the operation.

- For the “Read” and “Observe” operations, the LwM2M Client MUST obtain the access right for the LwM2M Server on each Object Instance according to Section 7.3.2.1 “Obtaining Access Right” and the LwM2M Client MUST retrieve all the Object Instances for which the LwM2M Server has “Read” access right; for each of these qualified Object Instances, the LwM2M Client MUST retrieve all the Resources except the Resources which do not support the “Read” operation. The LwM2M Client MUST then aggregate all the information individually produced by the operation on each of these Object Instances and send that to the LwM2M Server.
- For the “Write-Attributes” operation, the LwM2M Client MUST perform the operation.

7.2.5. Notify Operation Consideration

If the LwM2M Client needs to send a “Notify” operation containing an Object Instance or a Resource to the LwM2M Server, the LwM2M Client MUST check if the LwM2M Server is authorized for the “Read” operation. If the LwM2M Server is not authorized, the Client MUST NOT send the “Notify” operation.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No Prior Version

Table: A.1-1 Approved Version History

A.2 Draft Version History

Draft Versions for OMA-TS-LightWeightM2M-V1_1:

Date	Sections	Description
4 May 2016	n/a	First baseline.
5 Jul 2016	1,2,3,4,4.1,5,6,7,8,A-1	Incorporated CR:OMA-DM-LightweightM2M-2016-0076-CR_LWM2M_1.1_baseline_update, Editorial changes.
11 Jul 2016	7.1.1	Incorporated CR: OMA-DM-LightweightM2M-2016-0078-CR_LPWA_security.
20 Jul 2016	7.1.1, 7.1.2, 7.1.3, 7.1.4	Incorporated CR: OMA-DM-LightweightM2M-2016-0088-CR_Section_7_alignment.
2 Sep 2016	2, J	Incorporated CR: OMA-DM-LightweightM2M-2016-0105R01-CR_Annex_on_LWM2M_over_NB_IoT.
15 Sep 2017	Multiple	New baseline: OMA-TS-LightweightM2M_Core-V1_1-20170915-D.

Table: A.2-1 Draft 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 Bootstrap Interface

Item	Function	Reference	Requirement
LwM2M-BOOT-001-C-M	Support of at least one Bootstrap Mode	Section 5.1	
LwM2M-BOOT-002-C-O	Support of Factory Bootstrap Mode	Section 5.2.3.1	
LwM2M-BOOT-003-C-O	Support of Bootstrap from Smartcard	Section 5.2.3.2, Appendix F	LwM2M-BOOT-012C-O
LwM2M-BOOT-004-C-O	Support of Client Initiated Bootstrap	Section 5.2.3.3	
LwM2M-BOOT-005-C-O	Support of Server Initiated Bootstrap	Section 5.2.3.4	
LwM2M-BOOT-006-C-M	Support of LwM2M Server Bootstrap Information	Section 5.2.2	
LwM2M-BOOT-007-C-O	Support of LwM2M Bootstrap-Server Bootstrap Information	Section 5.2.2	
LwM2M-BOOT-008-C-M	Support of accepting Bootstrap Information transferred	Section 5.2.2	

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

Table: B.1.1-1 Bootstrap Interface

B.1.2 Client Registration

Item	Function	Reference	Requirement
LwM2M-CR-001-C-M	Support of “Register” operation	Section 5.3.1	
LwM2M-CR-002-C-M	Support of Endpoint Client Name parameter	Section 5.3.1	
LwM2M-CR-003-C-M	Support of Lifetime parameter	Section 5.3.1	
LwM2M-CR-004-C-M	Support of LwM2M Version parameter	Section 5.3.1	
LwM2M-CR-005-C-M	Support of Binding Mode parameter	Section 5.3.1, 5.3.1.1	
LwM2M-CR-006-C-O	Support of SMS Number parameter	Section 5.3.1	
LwM2M-CR-007-C-M	Support of Object and Object Instances parameter	Section 5.3.1	

LwM2M-CR-008-C-M	Support of “Update” operation	Section 5.3.2	
LwM2M-CR-009-C-O	Support of “De-register” operation	Section 5.3.3	
LwM2M-CR-010-C-O	Support of Updating Bootstrap Information from Smartcard at Register/Update	Section 5.3.2	(LwM2M-CR-001-C-M OR LwM2M-CR-008-C-M) AND LwM2M-BOOT-013-C-O AND (LwM2M-BOOT-003-C-O OR LwM2M-BOOT-011-C-O)
LwM2M-CR-0011-C-M	No Security Object (ID:0) and Security Object Instances in the parameters list	Section 5.3.1	
LwM2M-CR-0012-C-M	Support of Object Versioning in the Object and Object Instances parameter list	Section 5.3.1	

Table: B.1.2-1 Client Registration

B.1.3 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LwM2M-DMSE-001-C-M	Support of “Read” operation	Section 5.4.1	
LwM2M-DMSE-002-C-M	Support of “Discover” operation	Section 5.4.2	

LwM2M-DMSE-003-C-M	Support of “Write” operation	Section 5.4.3	
LwM2M-DMSE-004-C-M	Support of “Write-Attributes” operation	Section 5.4.4	
LwM2M-DMSE-005-C-O	Support of Minimum Period parameter	Section 5.4.4	
LwM2M-DMSE-006-C-O	Support of Maximum Period parameter	Section 5.4.4	
LwM2M-DMSE-007-C-O	Support of Greater Than parameter	Section 5.4.4	
LwM2M-DMSE-008-C-O	Support of Less Than parameter	Section 5.4.4	
LwM2M-DMSE-009-C-O	Support of Step parameter	Section 5.4.4	
LwM2M-DMSE-010-C-O	Support of Cancel parameter	Section 5.4.4	
LwM2M-DMSE-011-C-M	Support of “Execute” operation	Section 5.4.5	
LwM2M-DMSE-012-C-M	Support of “Create” operation	Section 5.4.6	
LwM2M-DMSE-013-C-M	Support of “Delete” operation	Section 5.4.7	

Table: B.1.3-1 Device Management and Service Enablement Interface

B.1.4 Information Reporting

Item	Function	Reference	Requirement
LwM2M-IR-001-C-M	Support of “Observe” operation	Section 5.5.1	
LwM2M-IR-002-C-M	Support of “Notify” operation	Section 5.5.2	
LwM2M-IR-003-C-M	Support of “Cancel Observation” operation	Section 5.5.3	

Table: B.1.4-1 Information Reporting

B.1.5 Data Format

Item	Function	Reference	Requirement
LwM2M-DF-001-C-O	Support of Plain Text format	Section 6.4, 6.4.1	
LwM2M-DF-002-C-O	Support of Opaque format	Section 6.4, 6.4.2	
LwM2M-DF-003-C-M	Support of TLV format	Section 6.4, 6.4.3	
LwM2M-DF-004-C-O	Support of JSON format	Section 6.4, 6.4.4	

Table: B.1.5-1 Data Format

B.1.6 Security

Item	Function	Reference	Requirement
------	----------	-----------	-------------

LwM2M-SEC-001-C-M	Support of at least one key mode	Section 7.1	LwM2M-SEC-002-C-O OR LwM2M-SEC-003-C-O OR LwM2M-SEC-004-C-O OR LwM2M-SEC-004-C-O
LwM2M-SEC-002-C-O	Support of Pre-Shared Keys mode	Section 7.1.7	
LwM2M-SEC-003-C-O	Support of Raw Public Key Certificates mode	Section 7.1.8	
LwM2M-SEC-004-C-O	Support of X.509 Certificates mode	Section 7.1.9	
LwM2M-SEC-005-C-O	Support of No Sec mode	Section 7.1.10	
LwM2M-SEC-006-C-O	Support of UDP Channel Security	Section 7.1	
LwM2M-SEC-007-C-O	Support of Smartcard Secure Channel	Section 7.1, Appendix G	LwM2M-SEC-009-C-O
LwM2M-SEC-008-C-O	Support of Access Control Mechanism	Section 7.3	

LwM2M-SEC-009-C-O	Smartcard Secure Channel using [GLOBALPLATFORM] [GP SCP03]		
-------------------	--	--	--

Table: B.1.6-1 Security

B.1.7 Mechanism

Item	Function	Reference	Requirement
LwM2M-MEC-001-C-O	Support of Queue Mode	Section 8.3	
LwM2M-MEC-002-C-M	Support of UDP Binding	Section 8.6.1	
LwM2M-MEC-003-C-O	Support of SMS Binding	Section 8.6.2	

Table: B.1.7-1 Mechanism

B.1.8 Objects

Item	Function	Reference	Requirement
LwM2M-OBJ-001-C-M	Support of LwM2M Security Object	Appendix E.1	
LwM2M-OBJ-002-C-M	Support of LwM2M Server Object	Appendix E.2	
LwM2M-OBJ-003-C-O	Support of Access Control Object	Appendix E.3	
LwM2M-OBJ-004-C-M	Support of Device Object	Appendix E.4	

LwM2M-Obj-005-C-O	Support of Connectivity Monitoring Object	Appendix E.5	
LwM2M-Obj-006-C-O	Support of Firmware Update Object	Appendix E.6	
LwM2M-Obj-007-C-O	Support of Location Object	Appendix E.7	
LwM2M-Obj-008-C-O	Support of Connectivity Statistics Object	Appendix E.8	

Table: B.1.8-1 Objects

B.2 SCR for LwM2M Server

B.2.1 Bootstrap Interface

Item	Function	Reference	Requirement
LwM2M-BOOT-005-S-M	Support of Server Initiated Bootstrap	Section 5.2.3.4	
LwM2M-BOOT-010-S-M	Support of Bootstrap Security	Section 5.2.5	

Table: B.2.1-1 Bootstrap Interface

B.2.2 Client Registration

Item	Function	Reference	Requirement
LwM2M-CR-001-S-M	Support of “Register” operation	Section 5.3.1	
LwM2M-CR-002-S-M	Support of Endpoint Client Name parameter	Section 5.3.1	

LwM2M-CR-003-S-M	Support of Lifetime parameter	Section 5.3.1	
LwM2M-CR-004-S-M	Support of LwM2M Version parameter	Section 5.3.1	
LwM2M-CR-005-S-M	Support of Binding Mode parameter	Section 5.3.1, 5.3.1.1	
LwM2M-CR-006-S-M	Support of SMS Number parameter	Section 5.3.1	
LwM2M-CR-007-S-M	Support of Object and Object Instances parameter	Section 5.3.1	
LwM2M-CR-008-S-M	Support of “Update” operation	Section 5.3.2	
LwM2M-CR-009-S-M	Support of “De-register” operation	Section 5.3.3	

Table: B.2.2-1 Client Registration

B.2.3 Device Management and Service Enablement Interface

Item	Function	Reference	Requirement
LwM2M-DMSE-001-S-M	Support of “Read” operation	Section 5.4.1	
LwM2M-DMSE-002-S-M	Support of “Discover” operation	Section 5.4.2	
LwM2M-DMSE-003-S-M	Support of “Write” operation	Section 5.4.3	
LwM2M-DMSE-004-S-M	Support of “Write-Attributes” operation	Section 5.4.4	
LwM2M-DMSE-005-S-M	Support of Minimum Period parameter	Section 5.4.4	
LwM2M-DMSE-006-S-M	Support of Maximum Period parameter	Section 5.4.4	
LwM2M-DMSE-007-S-M	Support of Greater Than parameter	Section 5.4.4	

LwM2M-DMSE-008-S-M	Support of Less Than parameter	Section 5.4.4	
LwM2M-DMSE-009-S-M	Support of Step parameter	Section 5.4.4	
LwM2M-DMSE-010-S-M	Support of “Execute” operation	Section 5.4.5	
LwM2M-DMSE-011-S-M	Support of “Create” operation	Section 5.4.6	
LwM2M-DMSE-012-S-M	Support of “Delete” operation	Section 5.4.7	

Table: B.2.3-1 Device Management and Service Enablement Interface

B.2.4 Information Reporting

Item	Function	Reference	Requirement
LwM2M-IR-001-S-M	Support of “Observe” operation	Section 5.5.1	
LwM2M-IR-002-S-M	Support of “Notify” operation	Section 5.5.2	
LwM2M-IR-003-S-M	Support of “Cancel Observation” operation	Section 5.5.3	

Table: B.2.4-1 Information Reporting

B.2.5 Data Format

Item	Function	Reference	Requirement
LwM2M-DF-001-S-M	Support of Plain Text format	Section 6.4, 6.4.1	

LwM2M-DF-002-S-M	Support of Opaque format	Section 6.4, 6.4.2	
LwM2M-DF-003-S-M	Support of TLV format	Section 6.4, 6.4.3	
LwM2M-DF-004-S-M	Support of JSON format	Section 6.4, 6.4.4	

Table: B.2.5-1 Data Format

B.2.6 Security

Item	Function	Reference	Requirement
LwM2M-SEC-002-S-M	Support of Pre-Shared Keys mode	Section 7.1.7	
LwM2M-SEC-003-S-M	Support of Raw Public Key Certificates mode	Section 7.1.8	
LwM2M-SEC-004-S-M	Support of X.509 Certificates mode	Section 7.1.9	
LwM2M-SEC-005-S-M	Support of No Sec mode	Section 7.1.10	
LwM2M-SEC-006-S-M	Support of UDP Channel Security	Section 7.1	

Table: B.2.6-1 Security

B.2.7 Mechanism

Item	Function	Reference	Requirement
LwM2M-MEC-001-S-M	Support of Queue Mode	Section 8.3	

LwM2M-MEC-002-S-M	Support of UDP Binding	Section 8.6.1	
LwM2M-MEC-003-S-O	Support of SMS Binding	Section 8.6.2	

Table: B.2.7-1 Mechanism

B.2.8 Objects

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

Table: B.2.8-1 Objects

Appendix C. Data Types (Normative)

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

Data Type	Description	Text Format	TLV Format
String	A UTF-8 string, the minimum and/or maximum length of the String MAY be defined.	Represented as a UTF-8 string.	Represented as a UTF-8 string of Length bytes.
Integer	An 8, 16, 32 or 64-bit signed integer. The valid range of the value for a Resource SHOULD be defined. This data type is also used for the purpose of enumeration.	<p>Represented as an ASCII signed integer.</p> <p>For example, the integer value -750 results in the 4 characters/byte long ASCII string “-750”.</p>	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).

Unsigned Integer	<p>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 be used as a bitmask whereby bit positions range from 0 to $\text{sizeof}(\text{unsigned integer})-1$. 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 $2^{\text{bit position}}$. The following example shows an 8-bit unsigned integer value illustrating a bitmask with "00010010" whereby bit(1) and bit(4) are set resulting in the unsigned integer value 18 (i.e., 2^1+2^4).</p>	<p>Represented as an ASCII unsigned integer.</p> <p>For example, the unsigned integer value 18 results in a 2-character/byte long ASCII string "18".</p>	<p>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).</p>
Float	<p>A 32 or 64-bit floating point value. The valid range of the value for a Resource SHOULD be defined.</p>	<p>Represented as an ASCII signed numeric representation.</p> <p>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.</p>	<p>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).</p>

Boolean	An 8 bit unsigned integer with the value 0 for False and the value 1 for True.	Represented as the ASCII value 0 or 1.	Represented as an 8 bit unsigned Integer with value 0, or 1. The Length of a Boolean value MUST always be 1 byte.
Opaque	A sequence of binary octets, the minimum and/or maximum length of the String MAY be defined.	Represented as a Base64 encoding of the binary data [RFC4648]. For example, the sequence of bytes (in hex notation) {0x01, 0x02, 0x03, 0x04, 0x05} converts to the ASCII string "AQIDBAU=" in Base64 encoding.	Represented as a sequence of binary data of Length bytes.
Time	Unix Time. A signed integer representing the number of seconds since Jan 1 st , 1970 in the UTC time zone.	Represented as an ASCII integer. For example, 1476186613 seconds since Jan 01 1970, which represents Tuesday, 11-Oct-16 11:50:13 UTC, are represented as the ASCII string "1476186613", which has 10 characters/bytes.	Same representation as Integer.

Objlnk	<p>Object Link. The object link is used to refer an Instance of a given Object. An Object link value is composed of two concatenated 16 bit unsigned integers following the Network Byte Order convention. The Most Significant Half-word is an Object ID, the Least Significant Half-word is an Object Instance ID.</p> <p>An Object Link referencing no Object Instance will contain the concatenation of 2 MAX-ID values (null link).</p>	<p>Represented as a UTF-8 string containing 2 16-bits ASCII integers separated by a ':' ASCII character.</p>	<p>Same representation as two 16 bit unsigned integers one beside the other. The first one represents the Object ID, and the second one represents the Object Instance ID. This value is always 4 bytes long.</p>
Corelnk	<p>CoRE Link. A link is used to refer to Resources on a LWM2M Client and their attributes as specified in [RFC6690].</p>	<p>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: < /3303/0/5700 >;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.</p>	<p>Same representation as String.</p>
none	<p>No specific data type affected to that resource: it exclusively concerns Executable Resource.</p>	<p>Not applicable</p>	<p>Not applicable</p>

Table: C.-1 Data Types

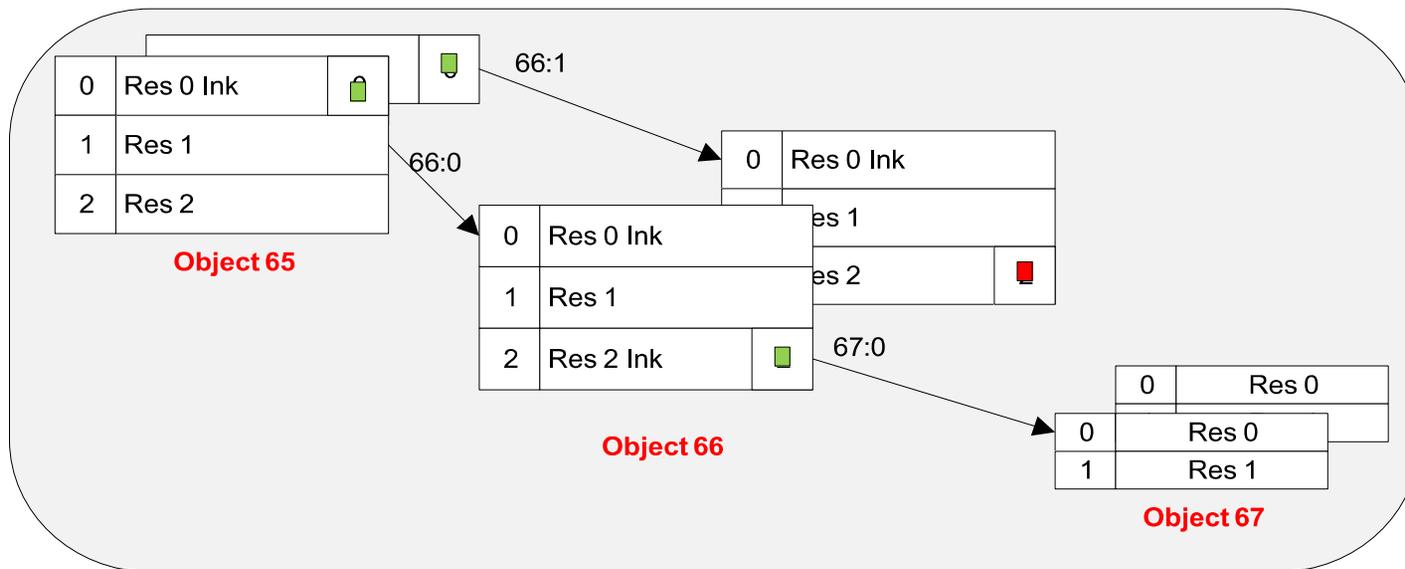


Figure: C.-1 Object link Resource simple illustration

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

Table: D.1-1 Object definition

- Name: specifies the Object name.
- Object ID: specifies the Object ID.
- Instances: indicates whether this Object supports multiple Object Instances or not. If this field is “Multiple” then the number of Object Instance can be from 0 to many. If this field is “Single” then the number of Object Instance can be from 0 to 1. If the Object field “Mandatory” is “Mandatory” and the Object field “Instances” is “Single” then, the number of Object Instance MUST be 1.
- Mandatory: if this field is “Mandatory”, then the LwM2M Client MUST support this Object. If this field is “Optional”, then the LwM2M Client SHOULD support this Object.
- Object URN: specifies the Object URN. The format of the Object URN is “urn:oma:lwm2m:{oma,ext,x}:{Object ID}[:{version}]” and {} part means that those values are variable and filled with real value. For example, Object URN of LwM2M Server Object is “urn:oma:lwm2m:oma:1”. The “version” field, follows the rules specified in Section 6.2 related to Object Versioning Policy.

Resource definition:

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description

0	Resource Name	R (Read), W (Write), E (Execute)	Multiple/Single	Mandatory/Optional	String, Integer, Float, Boolean, Opaque, Time, Objlnk none	If any	If any	Description
---	---------------	---	-----------------	--------------------	---	--------	--------	-------------

Table: D.1-2 Resource definition

- ID: specifies the Resource ID which is unique within Object.
- Name: specifies the Resource name.
- Operations: indicates which operations the Resource supports in the “Device Management & Service Enablement” Interface. This field can be set to a combination of R (Read, Observe, Discover, Write-Attributes), and W (Write), or can be set to E (Execute); Executable Operation is exclusive regarding the two others (R, W). This field may also have an empty value, which means that this field is not allowed to be accessed via “Device Management & Service Enablement” Interface but allowed to be accessed via “Bootstrap” Interface.
- Instances: indicates whether this Resource supports multiple Resource Instances or not. If this field is “Multiple” then the number of Resource Instance can be from 0 to many. If this field is “Single” then the number of Resource Instance can be from 0 to 1. If the Resource field “Mandatory” is “Mandatory” and the field “Instances” of the Resource is “Single” then, the number of Resource Instance MUST be 1. Resource which supports “Execute” operation MUST have “Single” as value of the “Instances” field.
- Mandatory: if this field is “Mandatory”, then any Instance of the Object that Resource belongs to, MUST instantiate such a Resource (refer Section 6.1, Resource Model). If this field is “Optional”, then this Resource MAY be omitted from some - or even all - Instances of the Object that Resource belongs to.
- Type: Data Type indicates the type of Resource value. Data Types used in this enabler are described in Appendix C Data Types. Resource which supports “Execute” operation MUST have no associated Data Type (none) encoded as

an empty value in Object DDF file.

- Range or Enumeration: this field limits the value of Resource.
- Units: specifies the unit of the Resource value.
- Description: specifies the Resource description.

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

This table provides the properties of arguments

Executable Resource Arguments Definition

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

Table: D.1-3 Executable Resource Arguments Definition

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

Executable Resource Arguments Definition

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

10	Create		
----	--------	--	--

Table: D.1-4 Example Executable Resource Arguments Definition

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. Other organizations and companies may define additional LwM2M according to the guidelines and template provided in Appendix D.

The following LwM2M Objects have been defined by OMA as part of LwM2M 1.0:

Object	Object ID
LwM2M Security	0
LwM2M Server	1
Access Control	2
Device	3
Connectivity Monitoring	4
Firmware	5
Location	6
Connectivity Statistics	7

Table: E.-1 LwM2M Objects defined by OMA LwM2M 1.0

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

E.1 LwM2M Object: LwM2M Security

Description

This LwM2M Object provides the keying material of a LwM2M Client appropriate to access a specified LwM2M Server.

One Object Instance SHOULD address a LwM2M Bootstrap-Server.

These LwM2M Object Resources MUST only be changed by a LwM2M Bootstrap-Server or Bootstrap from Smartcard and MUST NOT be accessible by any other LwM2M Server.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
LwM2M Security	0	Multiple	Mandatory	urn:oma:lwm2m:oma:0:1.1

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 UDP payload security mode is used 0: Pre-Shared Key mode 1: Raw Public Key mode 2: Certificate mode 3: NoSec mode 4: Certificate mode with EST
3	Public Key or Identity		Single	Mandatory	Opaque			Stores the LwM2M Client's Certificate or optional its certification path {including client certificate but not trust anchor} (Certificate mode), public key (RPK mode) or PSK Identity (PSK mode). The format is defined in Section E.1.1.
4	Server Public Key		Single	Mandatory	Opaque			Stores the LwM2M Server's, respectively LwM2M Bootstrap-Server's, Certificate or a trust anchor certificate suitable for path validation (Certificate mode), public key (RPK mode). The format is defined in Section E.1.1.
5	Secret Key		Single	Mandatory	Opaque			Stores the secret key or private key of the security mode. The format of the keying material is defined by the security mode in Section E.1.1. This Resource MUST only be changed by the bootstrap-server and MUST NOT be readable by any server.

6	SMS Security Mode		Single	Optional	Integer	0-255		<p>Determines which SMS security mode is used (see section 7.2)</p> <p>0: Reserved for future use</p> <p>1: DTLS mode (Device terminated) P mode assumed</p> <p>2: Secure Packet Structure mode (Smartcard terminated)</p> <p>3: NoSec mode</p> <p>4: Reserved mode (DTLS mode with multiplexing Security Association support)</p> <p>5-203 : Reserved for future use</p> <p>204-255: Proprietary modes</p>
7	SMS Binding Key Parameters		Single	Optional	Opaque	6 bytes		<p>Stores the KIC, KID, SPI and TAR. The format is defined in Section E.1.2.</p>
8	SMS Binding Secret Key(s)		Single	Optional	Opaque	16-32-48 bytes		<p>Stores the values of the key(s) for the SMS binding.</p> <p>This resource MUST only be changed by a bootstrap-server and MUST NOT be readable by any server.</p>

9	LwM2M Server SMS Number		Single	Optional	String		<p>MSISDN used by the LwM2M Client send messages to the LwM2M Server via the SMS binding.</p> <p>The LwM2M Client SHALL silently ignore any SMS originated from unknown MSISDN</p>
10	Short Server ID		Single	Optional	Integer	1-65534	<p>This identifier uniquely identifies each LwM2M Server configured for the LwM2M Client.</p> <p>This Resource MUST be set when the Bootstrap-Server Resource has false value.</p> <p>Specific ID:0 and ID:65535 values MUST NOT be used for identifying the LwM2M Server (Section 6.3).</p>
11	Client Hold Off Time		Single	Optional	Integer	s	<p>Relevant information for a Bootstrap Server only.</p> <p>The number of seconds to wait before initiating a Client Initiated Bootstrap once the LwM2M Client has determined it should initiate this bootstrap mode.</p> <p>In case client initiated bootstrap is supported by the LwM2M Client, this resource MUST be supported.</p>

12	Bootstrap-Server Account Timeout		Single	Optional	Integer		s	<p>The LwM2M Client MUST purge the LwM2M Bootstrap-Server Account after the timeout value given by this resource. The lowest timeout value is 0.</p> <p>If the value is set to 0, or if this resource is not instantiated, the Bootstrap-Server Account lifetime is infinite.</p>
13	Matching Type		Single	Optional	Integer	0-3		<p>The Matching Type Resource specifies how the certificate or raw public key in the Server Public Key is presented. Four values are currently defined:</p> <p>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.</p> <p>1: SHA-256 hash [RFC6234]</p> <p>2: SHA-384 hash [RFC6234]</p> <p>3: SHA-512 hash [RFC6234]</p>
14	SNI		Single	Optional	String			<p>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.</p>

15	Certificate Usage		Single	Optional	Integer	0-3		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	Integer		When this resource is present it instructs the DTLS/TLS client to propose the indicated ciphersuite(s) 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/parameters/tls-parameters.xhtml . For example, the TLS_PSK_WITH_AES_128_CCM_8 ciphersuite is represented with the following string "0xCo,0xA8". To form an integer value the two values are concatenated. In this example, the value is 0xc0a8 or 49320.
----	----------------------	--	----------	----------	---------	--	--

Table: E.1-2 LwM2M Object: LwM2M Security resource definitions

E.2 LwM2M Object: LwM2M Server

Description

This LwM2M Object provides the data related to a LwM2M Server. A Bootstrap-Server has no such an Object Instance associated to it.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
LwM2M Server	1	Multiple	Mandatory	urn:oma:lwm2m:oma:1:1.1

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 Section 5.3 Registration)
2	Default Minimum Period	RW	Single	Optional	Integer		s	The default value the LwM2M Client should use for the Minimum Period of an Observation in the absence of this parameter being included in an Observation. If this Resource doesn't exist, the default value is 0.
3	Default Maximum Period	RW	Single	Optional	Integer		s	The default value the LwM2M Client should use for the Maximum Period of an Observation in the absence of this parameter being included in an Observation.

4	Disable	E	Single	Optional				<p>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.</p>
5	Disable Timeout	RW	Single	Optional	Integer		s	<p>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).</p>

6	Notification Storing When Disabled or Offline	RW	Single	Mandatory	Boolean		<p>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.</p> <p>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.</p>
---	---	----	--------	-----------	---------	--	--

7	Binding	RW	Single	Mandatory	String	The possible values of Resource are listed in 5.3.1.1	This Resource defines the transport binding configured for the LwM2M Client. If the LwM2M Client supports the binding specified in this Resource, the LwM2M Client MUST use that transport for the Current Binding Mode.
8	Registration Update Trigger	E	Single	Mandatory			If this Resource is executed the LwM2M Client MUST perform an "Update" operation with this LwM2M Server using the relevant transport for the current Binding Mode.
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 to be used to communicate with this server.

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	Instances	Mandatory	Object URN
LwM2M Access Control	2	Multiple	Optional	urn:oma:lwm2m:oma:2:1.1

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

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Object ID	R	Single	Mandatory	Integer	1-65535		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		<p>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.</p> <p>The Resource Instance ID 0 is a specific ID, determining the ACL Instance which contains the default access rights.</p> <p>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.</p> <p>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.</p>

3	Access Control Owner	RW	Single	Mandatory	Integer	1-65535	<p>Short Server ID of a certain LwM2M Server; only such an LwM2M Server can manage the Resources of this Object Instance.</p> <p>The specific value MAX_ID=65535 means this Access Control Object Instance is created and modified during a Bootstrap phase only.</p>
---	----------------------	----	--------	-----------	---------	---------	---

Table: E.3-2 LwM2M Object: 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	Instances	Mandatory	Object URN
Device	3	Single	Mandatory	urn:oma:lwm2m:oma:3

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		<p>0 – DC power 1 – Internal Battery 2 – External Battery 4 – Power over Ethernet 5 – USB 6 – AC (Mains) power 7 – Solar</p> <p>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)</p>
7	Power Source Voltage	R	Multiple	Optional	Integer		mV	Present voltage for each Available Power Sources Resource Instance.
8	Power Source Current	R	Multiple	Optional	Integer		mA	Present current for each Available Power Source.
9	Battery Level	R	Single	Optional	Integer	0-100	%	Contains the current battery level as a percentage (with a range from 0 to 100). This value is only valid for the Device Internal Battery if present (one Available Power Sources Resource Instance is 1).

10	Memory Free	R	Single	Optional	Integer		KB	Estimated current available amount of storage space which can store data and software in the LwM2M Device (expressed in kilobytes).
11	Error Code	R	Multiple	Mandatory	Integer	0-8		<p>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</p> <p>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</p>

							<p>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.</p> <p>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.</p>
12	Reset Error Code	E	Single	Optional			<p>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.</p>

13	Current Time	RW	Single	Optional	Time			Current UNIX time of the LwM2M Client. The LwM2M Client should be responsible to increase this time value as every second elapses. The LwM2M Server is able to write this Resource to make the LwM2M Client synchronized with the LwM2M Server.
14	UTC Offset	RW	Single	Optional	String			Indicates the UTC offset currently in effect for this LwM2M Device. UTC+X [ISO 8601].
15	Timezone	RW	Single	Optional	String			Indicates in which time zone the LwM2M Device is located, in IANA Timezone (TZ) database format.
16	Supported Binding and Modes	R	Single	Mandatory	String			Indicates which bindings and modes are supported in the LwM2M Client. The possible values of Resource are combination of "U" or "UQ" and "S" or "SQ".

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). See Battery Status table below.

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

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	<p>Indicates the network bearer used for the current LwM2M communication session from the below network bearer list.</p> <p>0~20 are Cellular Bearers 0: GSM cellular network 1: TD-SCDMA cellular network 2: WCDMA cellular network 3: CDMA2000 cellular network 4: WiMAX cellular network 5: LTE-TDD cellular network 6: LTE-FDD cellular network 7: NB-IoT 8~20: Reserved for other type cellular network</p> <p>21~40 are Wireless Bearers 21: WLAN network 22: Bluetooth network 23: IEEE 802.15.4 network 24~40: Reserved for other type local wireless network</p> <p>41~50 are Wireline Bearers 41: Ethernet 42: DSL 43: PLC 44~50: reserved for others type wireline networks.</p>
---	----------------	---	--------	-----------	---------	--

1	Available Network Bearer	R	Multiple	Mandatory	Integer			Indicates list of current available network bearer. Each Resource Instance has a value from the network bearer list.
2	Radio Signal Strength	R	Single	Mandatory	Integer		dBm	<p>This node contains the average value of the received signal strength indication used in the current network bearer (as indicated by Resource 0 of this Object).</p> <p>For the following network bearers the signal strength parameters indicated below are represented by this resource:</p> <p>GSM: RSSI UMTS: RSCP LTE: RSRP NB-IoT: NRSRP</p> <p>For the following network bearers the signal strength parameters indicated below are represented by this resource:</p> <p>GSM: RSSI UMTS: RSCP LTE: RSRP</p> <p>For more details on Network Measurement Report, refer to</p>

								the appropriate Cellular or Wireless Network standards. (e.g., for LTE Cellular Network refer to ETSI TS 36.133 specification).
3	Link Quality	R	Single	Optional	Integer			This contains received link quality e.g. LQI for IEEE 802.15.4 (range 0..255), RxQual Downlink for GSM (range 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 average utilization of the link to the next-hop IP router in %.
7	APN	R	Multiple	Optional	String			Access Point Name in case Network Bearer Resource is a Cellular Network.
8	Cell ID	R	Single	Optional	Integer			Serving Cell ID in case Network Bearer Resource is a Cellular Network. As specified in TS [3GPP_TS_23.003] and in [3GPP_TS_24.008]. Range (0...65535) in GSM/EDGE UTRAN Cell ID has a length of 28 bits. Cell Identity in WCDMA/TD-SCDMA. Range: (0..268435455). LTE Cell ID has a length of 28 bits. Parameter definitions in [3GPP_TS_25.331].
9	SMNC	R	Single	Optional	Integer			Serving Mobile Network Code. In case Network Bearer Resource has 0(cellular network). Range (0...999). As specified in TS [3GPP_TS_23.003].

10	SMCC	R	Single	Optional	Integer		Serving Mobile Country Code. In case Network Bearer Resource has 0 (cellular network). Range (0...999). As specified in TS [3GPP_TS_23.003].
----	------	---	--------	----------	---------	--	--

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 firmware package, updating firmware, and performing actions after updating firmware. The firmware update MAY require to reboot the device; it will depend on a number of factors, such as the operating system architecture and the extent of the updated software.

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

A LwM2M Server may also instruct a LwM2M Client to fetch a firmware image from a dedicated server (instead of pushing firmware image 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.

Once a new firmware image is successfully installed in the Device, if the Resource “Firmware Version” ID:3 is available in the Instance of the Device Object (ID:3) the LwM2M Client MUST update such a Resource accordingly to the new firmware image version.

Note : PkgName (ID:6) and PkgVersion (ID:7) Resources contain optional information handled by the Firmware Update functionality for its own purpose and have no direct correlation with the “Firmware Version” ID:3 of Device Object ID:3.

Object definition

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

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 bytes	<p>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.</p> <p>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.</p>
---	-------------	----	--------	-----------	--------	-------------	--

2	Update	E	Single	Mandatory	none		<p>Updates firmware by using the firmware package stored in Package, or, by using the firmware downloaded from the Package URI.</p> <p>This Resource is only executable when the value of the State Resource is Downloaded.</p>
3	State	R	Single	Mandatory	Integer	0-3	<p>Indicates current state with respect to this firmware update. This value is set by the LwM2M Client.</p> <p>0: Idle (before downloading or after successful updating)</p> <p>1: Downloading (The data sequence is on the way)</p> <p>2: Downloaded</p> <p>3: Updating</p> <p>If writing the firmware package to Package Resource is done, or, if the device has downloaded the firmware package from the Package URI the state changes to Downloaded.</p> <p>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</p>

							<p>value is set to Idle and the Update Result Resource value is set to 0.</p> <p>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.</p> <p>Firmware Update mechanisms are illustrated below in Figure 20 of the LwM2M version 1.0 specification.</p>
5	Update Result	R	Single	Mandatory	Integer	0-9	<p>Contains the result of downloading or updating the firmware</p> <p>0: Initial value. Once the updating process is initiated (Download /Update), this Resource MUST be reset to Initial value.</p> <p>1: Firmware updated successfully,</p> <p>2: Not enough flash memory</p>

								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 bytes		Name of the Firmware Package
7	PkgVersion	R	Single	Optional	String	0-255 bytes		Version of the Firmware package

8	Firmware Update Protocol Support	R	Multiple	Optional	Integer		<p>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.</p> <p>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.</p> <p>The following values are defined by this version of the specification:</p> <p>0 – CoAP (as defined in RFC 7252) with the additional support for block-wise transfer. CoAP is the default setting.</p> <p>1 – CoAPS (as defined in RFC 7252) with the additional support for block-wise transfer</p>
---	----------------------------------	---	----------	----------	---------	--	--

							<p>2 – HTTP 1.1 (as defined in RFC 7230)</p> <p>3 – HTTPS 1.1 (as defined in RFC 7230)</p> <p>Additional values MAY be defined in the future. Any value not understood by the LwM2M Server MUST be ignored.</p>
9	Firmware Update Delivery Method	R	Single	Mandatory	Integer		<p>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.</p> <p>0 – Pull only</p> <p>1 – Push only</p> <p>2 – Both. In this case the LwM2M Server MAY choose the preferred mechanism for conveying the firmware image to the LwM2M Client.</p>

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

E.6.1 Firmware Update State Machine

Figure 20 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.

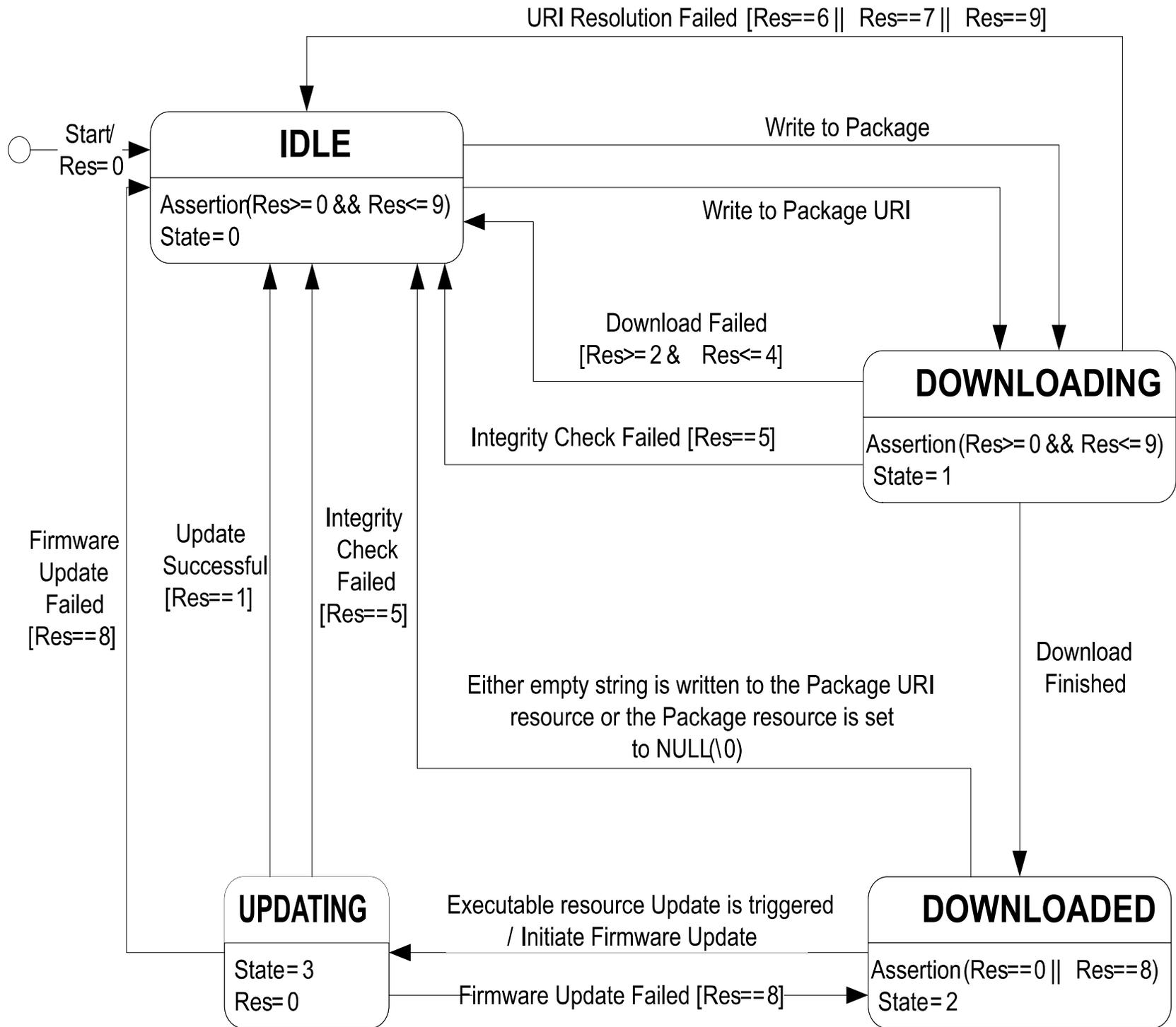


Figure: E.6.1-1 Firmware Update Mechanisms

E.6.2 Examples

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

```

LwM2M Server                                     LwM2M Client
|                                                 |
| CON [MID=1], POST, /5/0/0, 1:0/1/128 -----> |
| <----- ACK [MID=1], 2.31 Continue, 1:0/1/128 |
| CON [MID=2], POST, /5/0/0, 1:1/1/128 -----> |
| <----- ACK [MID=2], 2.31 Continue, 1:1/1/128 |
|           ... 637 exchanges ...                |
| CON [MID=640], POST, /5/0/0, 1:639/0/128 -----> |
| <----- ACK [MID=640], 2.04 Changed, 1:639/0/128 |
|                                                 |

```

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

The second example shown in Figure 31 illustrates the case where the client was provided with a URI by the LwM2M Server (using the Package URI resource) and therefore fetches the firmware image from the indicated server. Note that

only the retrieval of the firmware image from the LwM2M Server is shown in Figure 31 and not the initial configuration of the Package URI.

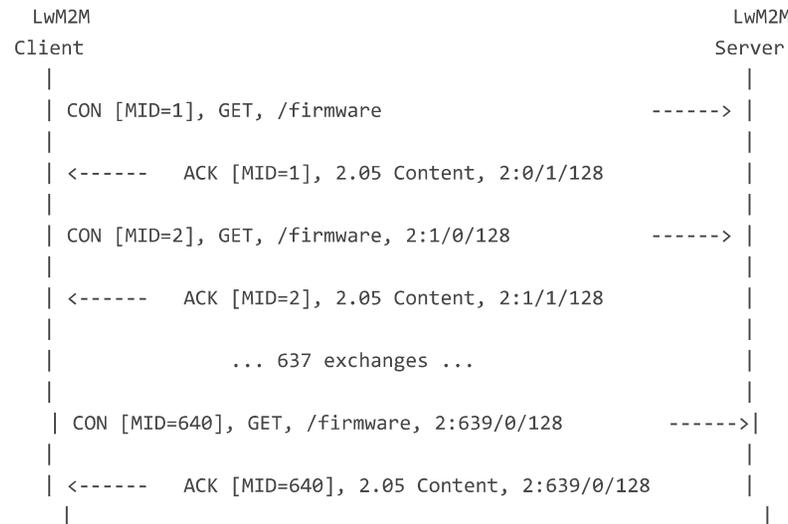


Figure: E.6.2-2 Example of a client fetching a firmware image

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

The LwM2M Location Object provides the ability to express a point (in 2d, and in 3d), a circle (in 2d) and a sphere (in 3d). This location information is used to indicate where the LwM2M Client is located at the indicated point in time. A point is used when there is no known uncertainty and it forms part of a number of other geometries. A point may be specified

using either world geodetic system (WGS) 84 (latitude, longitude) or WGS 84 (latitude, longitude, altitude). The circular area is used for coordinates in two-dimensional coordinate reference systems (CRSs) to describe uncertainty about a point. It is specified using a two-dimensional CRS (using latitude, longitude). The sphere is a volume that provides the same information as a circle in three dimensions. It is specified using a three-dimensional CRS (using latitude, longitude, and altitude). The use of the world geodetic system 1984 (WGS84) coordinate reference system and the usage of European petroleum survey group (EPSG) code 4326 for two-dimensional (2d) shape representations and EPSG 4979 for three-dimensional (3d) volume representations is mandated. For the circle and the sphere a confidence value of 95% is assumed [RFC7459]. Finally, the ability to convey information about moving objects is enabled by expression of speed and velocity.

Object definition

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

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

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
o	Latitude	R	Single	Mandatory	Float		Deg	The decimal notation of latitude, e.g., -43.5723 [World Geodetic System 1984].

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

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

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		Kilo-Bytes	Indicate the total amount of IP data transmitted during the collection period.

3	Rx Data	R	Single	Optional	Integer		Kilo-Bytes	Indicate the total amount of IP data received during the collection period.
4	Max Message Size	R	Single	Optional	Integer		Byte	The maximum IP message size that is used during the collection period.
5	Average Message Size	R	Single	Optional	Integer		Byte	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.
7	Stop	E	Single	Mandatory				Stop collecting information, but do not reset resources 0-5.
8	Collection Period	RW	Single	Optional	Integer		Seconds	The default collection period in seconds. The value 0 indicates that the collection period is not set.

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.

Appendix F. Example LwM2M Client (Informative)

This appendix defines an example LwM2M Client for a simple imaginary device with a Cellular interface including instantiated Objects and their values, which is used throughout this specification in examples. The example client has the Endpoint Name “example-client”. The example device has two Server Objects (it is configured to register with two different LwM2M Servers), five accompanying Access Control Object Instances for those servers, a Device Object, a Connectivity Monitoring Object for a Cellular interface and a Firmware Update Object with no instance. The first Server controls the access control rights for both servers.

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

Object	Object ID	Object Instance ID
LwM2M Security Object[0]	0	0
LwM2M Security Object[1]	0	1
LwM2M Security Object[2]	0	2
LwM2M Server Object [1]	1	0
LwM2M Server Object [2]	1	1
Access Control Object [0]	2	0
Access Control Object [1]	2	1

Access Control Object [2]	2	2
Access Control Object [3]	2	3
Access Control Object [4]	2	4
Device Object	3	0
Connectivity Monitoring Object	4	0
Firmware Update Object	5	-

Table: 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		b313cc22-f969-42ec-ad42	Example of a PSK Identity string
Server Public Key	4			Unused in PSK mode
Secret Key	5		e129791359950cbb1c8c3c582913b551	128-bit random sequence in hex encoding for use as a PSK.

Client Hold Off Time	11		3600	
----------------------	----	--	------	--

Table: 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 3d020106082a8648ce3d03010703420004a09dcb 1bc739e7f19afa9adcb1944968bfba73efcec29b 50486eb44d1b29c193449970a3736830cb2bd5d7 ec05d2bd1fc07b6df5e48b54ce77a6a7229c3a91 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 3d020106082a8648ce3d0301070342000482c773 f378d30c2783a48eb811b96cf6a90694a8564f1e 7f6d366152f1169895of6f7c40cda585334f8377 50a102f5bf25508ceb9e55dfcof12a455199a4c9 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 a16495748e146fcb5370b8e96d292ced5567a8fa e55a22bb67d91651b8a00a06082a8648ce3d0301 07a14403420004a09dcb1bc739e7f19afa9adcb1 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:

```
'3059301306072a8648ce3d020106082a8648ce3d0301070342000482c773f378d30c2783a48eb811b96cf6a90694a8564f1e7f6d366152f11698950f6f7c40cda585334f837750a102f5bf25508ceb9e55dfcof12a455199a4c960'
```

Then, the hex data needs to be converted 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		UQ	UDP with Queuing 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	0b00000000000001111	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	0b00000000000001111	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	0b00000000000001111	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	0b00000000000000001	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	ob0000000000000001	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	ob0000000000000001	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	0b00000000000010000	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 2 nd , 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
Link Quality	3		2	RxQual Downlink
IP Addresses	4	0	192.168.0.100	
Router IP Addresses	5	0	192.168.1.1	
Link Utilization	6		5	%
APN	7	0	internet	

Table: 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 Appendix G.3.2 and [PKCS#15].
- The EF (DODF-bootstrap) is described in Appendix G.3.3 and [PKCS#15].
- The EF (LwM2M Bootstrap data) is described in Appendix G.3.4.

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 G3.3) 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)

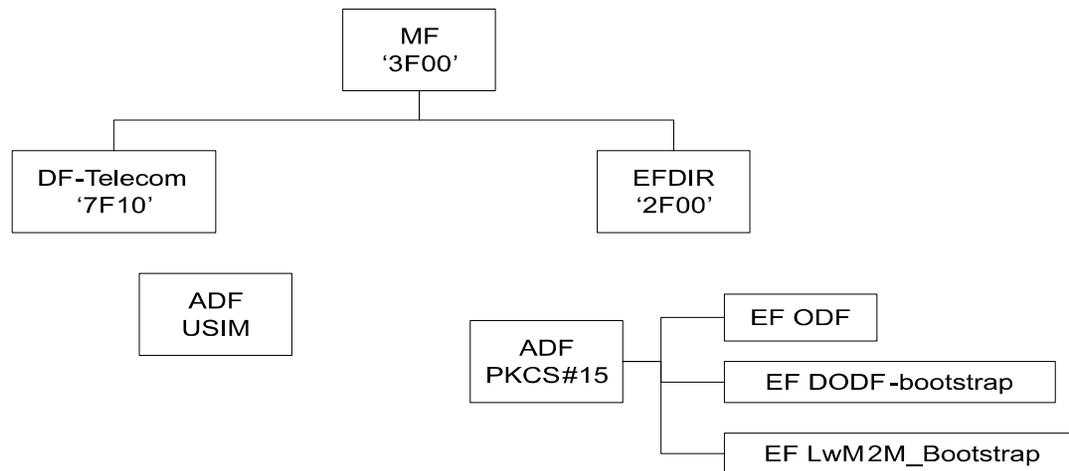


Figure: G.2.2-1 Example of a UICC File Structure embedding a specific PKCS\#15 file structure containing LwM2M Bootstrap data location

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)

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.

- 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 G.2.5 & [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 contain 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 Appendix G.2) 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 Appendix G.2) 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 (§6.4.3), this section will only detailed how to store a collection of LwM2M Objects in this EF LwM2M_Bootstrap file; each Object is coded with a header containing a LwM2M Object ID and its Object Version coded in one or 2 Bytes, a LwM2M-TLV coding the Object Instances as payload, and a length being the size in bytes of this payload (LwM2M-TLV of the Object Instances). 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 §6.6.4.3
```

```
HWORD ::= %x00-FFFF
```

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

```
MAJOR_VERSION ::= %x01-FF
```

```
MINOR_VERSION ::= %x00-FF
```

In reading and processing the data of this file, the LwM2M Client is then able to be configured with the Bootstrap Information and thus to access the LwM2M Server(s).

Appendix H. Secure channel between Smartcard and LwM2M Device Storage for secure Bootstrap Data provisioning (Normative)

During LwM2M Bootstrap procedure, sensitive data have to be provisioned in LwM2M Device.

When Bootstrap information comes from Smartcard, a Secure Channel SHOULD be established 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 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 (0xA0000004120002000000000000) 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 24):

- 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 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 Annex G.

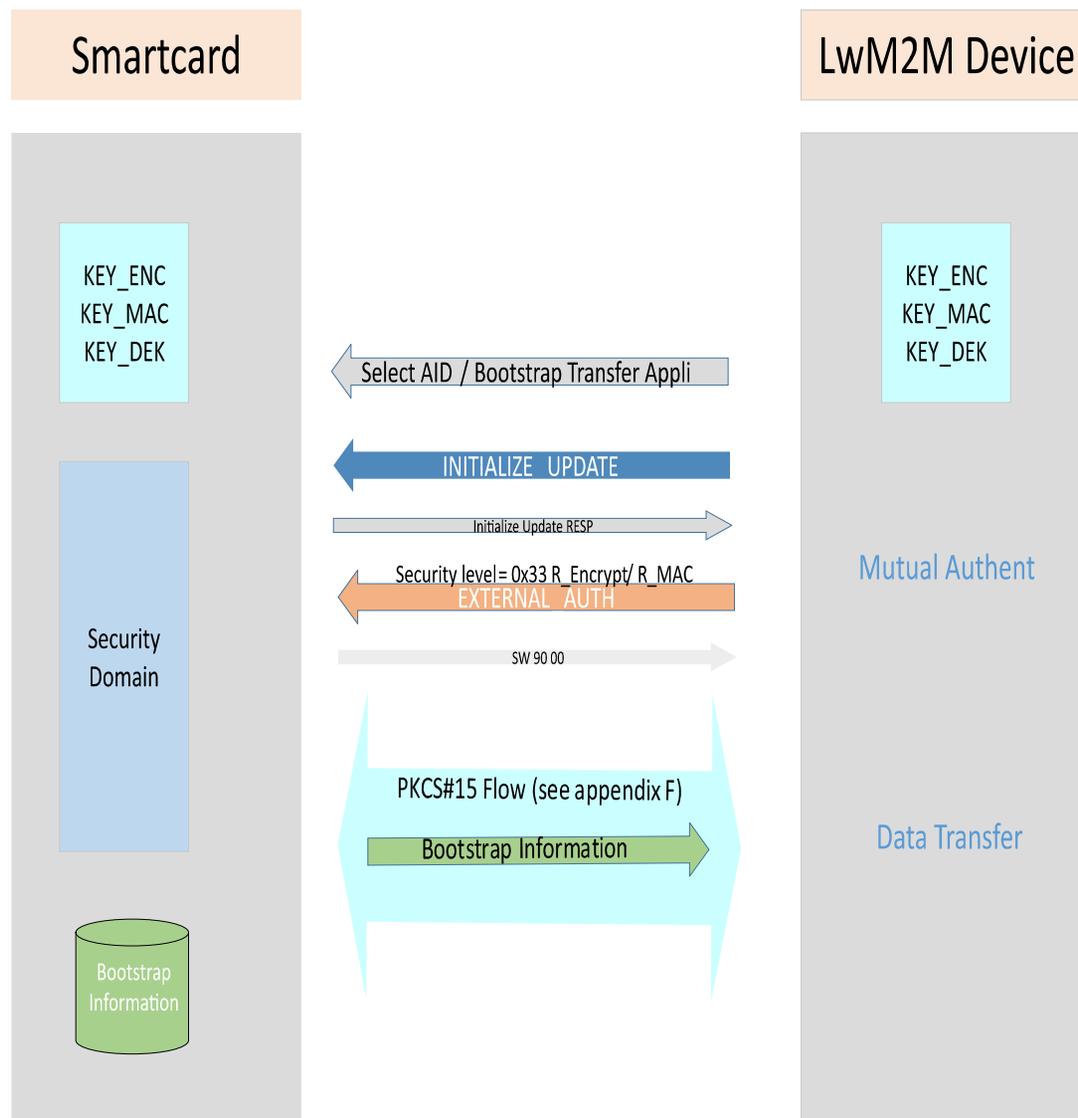


Figure: H.-1 Bootstrap Information transfer from Smartcard to LwM2M Device using Secure Channel according to [GLOBALPLATFORM] [GP SCP03] [GP AMD_A]

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.

Appendix I. Media types

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

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

Type name: application

Subtype name: vnd.oma.lwm2m+tlv

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations:

OMA LwM2M data is passive, meaning it does not contain executable or active content which may represent a security threat. The OMA LwM2M TLV format does not contain fields which are confidential. The usage of the

LwM2M TLV format may be vulnerable to attacks modifying or spoofing the content of this format. The OMA LwM2M protocol uses source authentication and integrity protection.

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

Interoperability considerations:

This content type carries OMA LwM2M data model serialization within the scope of the OMA LwM2M enabler. The OMA LwM2M enabler specification includes static conformance requirements for this content.

Published specification:

OMA LwM2M 1.0 Technical Specification – especially section 6.4.3. Available from <http://www.openmobilealliance.org>

Applications, which use this media type:

OMA LwM2M

Fragment identifier considerations: none

Additional information:

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

Intended usage: Limited use. Only for usage with OMA LwM2M, which meet the semantics given in the mentioned specification.

Restriction on usage: no

Person & email address to contact for further information:

John Mudge

helpdesk@omaorg.org

Author/Change controller:

Open Mobile Naming Authority (OMNA)

OMA-OMNA@mail.openmobilealliance.org

Provisional registration (standards tree only): N/A

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

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

Type name: application

Subtype name: vnd.oma.lwm2m+json

Required parameters: none

Optional parameters: none

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

See RFC7159.

Security considerations:

OMA LwM2M data is passive, meaning it does not contain executable or active content which may represent a security threat. The OMA LwM2M JSON format does not contain fields which are confidential. The usage of the LwM2M JSON format may be vulnerable to attacks modifying or spoofing the content of this format. The OMA LwM2M protocol uses source authentication and integrity protection.

Interoperability considerations:

This content type carries OMA LwM2M data model serialization within the scope of the OMA LwM2M enabler. The OMA LwM2M enabler specification includes static conformance requirements for this content.

Published specification:

OMA LwM2M 1.0 Technical Specification – especially section 6.4.4. Available from <http://www.openmobilealliance.org>

Applications which use this media type:

OMA LwM2M

Fragment identifier considerations: none

Additional information:

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

Intended usage: Limited use. Only for usage with OMA LwM2M, which meet the semantics given in the mentioned specification.

Restriction on usage: no

Person & email address to contact for further information:

John Mudge

helpdesk@omaorg.org

Author/Change controller:

Open Mobile Naming Authority (OMNA)

OMA-OMNA@mail.openmobilealliance.org

Provisional registration (standards tree only): N/A

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

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

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

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 6.2 “Object Versioning” in this document.

In addition:

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<LWM2M xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://openmobilealliance.org/tech/profiles/LWM2M.xsd" >
  <Object ObjectType="MDefinition">
    <Name>MyDevice</Name>
    <Description1><![CDATA[This LWM2M Object is my device]]></Description1>
    <ObjectID>44</ObjectID>
    <LWM2MVersion>1.1</LWM2MVersion>
  </Object>
</LWM2M>
```

```

    <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

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="LWM2M">

    <xs:complexType>

      <xs:sequence>

        <xs:element minOccurs="1" maxOccurs="unbounded" name="Object">

          <xs:complexType>

            <xs:sequence>

```

```
<xs:element name="Name" type="xs:string" />

<xs:element name="Description1" type="xs:string" />

<xs:element name="ObjectID" type="xs:unsignedShort" />

<xs:element name="ObjectURN" type="xs:string" />

<xs:element name="LWM2MVersion" type="xs:string" minOccurs="0" />

<xs:element name="ObjectVersion" type="xs:string" minOccurs="0" />

<xs:element name="MultipleInstances">

  <xs:simpleType>

    <xs:restriction base="xs:string">

      <xs:enumeration value="Multiple" />

      <xs:enumeration value="Single" />

    </xs:restriction>

  </xs:simpleType>

</xs:element>

<xs:element name="Mandatory">

  <xs:simpleType>

    <xs:restriction base="xs:string">

      <xs:enumeration value="Mandatory" />

      <xs:enumeration value="Optional" />

    </xs:restriction>

  </xs:simpleType>

</xs:element>

<xs:element name="Resources">

  <xs:complexType>
```

```
<xs:sequence>

  <xs:element maxOccurs="unbounded" name="Item">

    <xs:complexType>

      <xs:sequence>

        <xs:element name="Name" type="xs:string" />

        <xs:element name="Operations">

          <xs:simpleType>

            <xs:restriction base="xs:string">

              <xs:enumeration value="R" />

              <xs:enumeration value="W" />

              <xs:enumeration value="RW" />

              <xs:enumeration value="E" />

              <xs:enumeration value="" />

            </xs:restriction>

          </xs:simpleType>

        </xs:element>

        <xs:element name="MultipleInstances">

          <xs:simpleType>

            <xs:restriction base="xs:string">

              <xs:enumeration value="Multiple" />

              <xs:enumeration value="Single" />

            </xs:restriction>

          </xs:simpleType>

        </xs:element>

      </xs:sequence>

    </xs:complexType>

  </xs:element>

</xs:sequence>
```

```
</xs:element>

<xs:element name="Mandatory">

  <xs:simpleType>

    <xs:restriction base="xs:string">

      <xs:enumeration value="Mandatory" />

      <xs:enumeration value="Optional" />

    </xs:restriction>

  </xs:simpleType>

</xs:element>

<xs:element name="Type">

  <xs:simpleType>

    <xs:restriction base="xs:string">

      <xs:enumeration value="String" />

      <xs:enumeration value="Integer" />

      <xs:enumeration value="Float" />

      <xs:enumeration value="Boolean" />

      <xs:enumeration value="Opaque" />

      <xs:enumeration value="Time" />

      <xs:enumeration value="Objlnk" />

      <xs:enumeration value="Unsigned Integer"/>

      <xs:enumeration value="Corelnk"/>

      <xs:enumeration value="" />

    </xs:restriction>

  </xs:simpleType>
```

```
</xs:element>

<xs:element name="RangeEnumeration" type="xs:string" />

<xs:element name="Units" type="xs:string" />

<xs:element name="Description" type="xs:string" />

</xs:sequence>

<xs:attribute name="ID" type="xs:unsignedShort" use="required" />

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="Description2" type="xs:string" />

</xs:sequence>

<xs:attribute name="ObjectType" type="xs:string" use="required" />

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>
```

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	<p>PDU length:14 CoAP Version: 1 Message Type: Confirmable Token length: 4 Code: 0.02 POST Message ID: 34488 Token length: 4 Value: 0x301e65ca</p> <p>OPTION (0/3) Option number (delta): 11(11) Name: URI_PATH Value length: 1 Value: "1"</p> <p>OPTION (1/3) Option number (delta): 11(0) Name: URI_PATH Value length: 1 Value: "0"</p> <p>OPTION (2/3) Option number (delta): 11(0) Name: URI_PATH Value length: 1 Value: "8" No payload.</p>
---	--

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

Appendix M. LWM2M over NB-IoT

M.1 Introduction

Note: The 3GPP vocabulary and abbreviations used in the Annex are explained in [3GPP-TS_21.905]. 3GPP has specified Narrow-Band IoT (NB-IoT) as part of their Release 13. NB-IoT includes solutions for support of infrequent data transmission via user plane and via control plane (=data transfer via MME). The user plane solution includes IP data and SMS support. The control plane solution includes IP data, non-IP data and SMS support. Main focus of this Annex is on considerations and current limitations when running CoAP over the non-IP mode of NB-IoT. The figure below shows the 3GPP NB-IoT architecture as in [3GPP-TR_23.720] and the LWM2M protocol stack. The shown C-SGN combines the functionality of MME, S-GW, and P-GW.

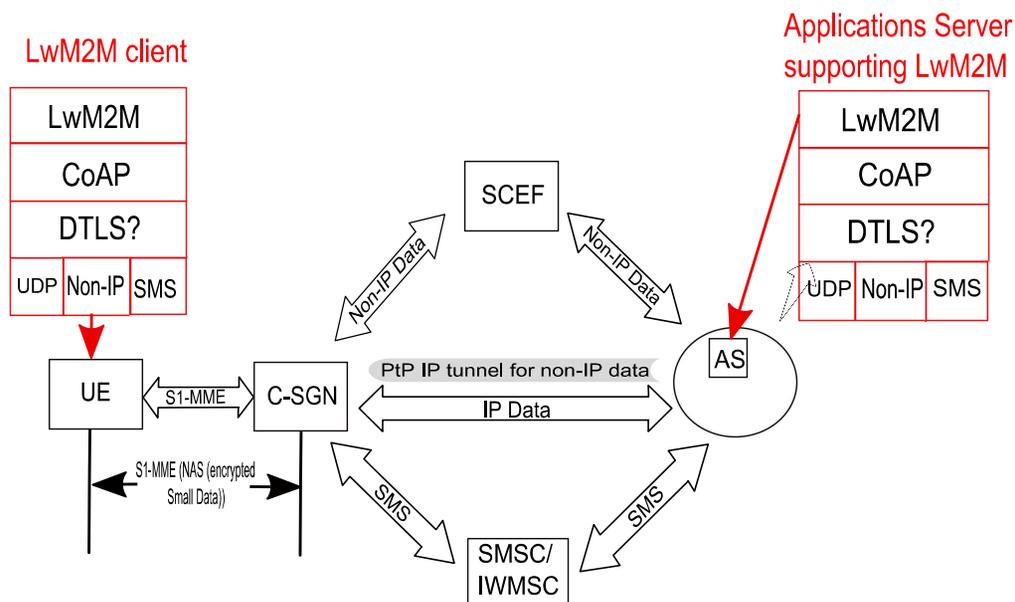


Figure: M.1-1 3GPP NB-IoT architecture as in [3GPP-TR_23.720] and the LWM2M protocol stack

As can be seen from the above figure 3GPP defines two transmission paths for NIDD (Non-IP Data Delivery):

1. via PtP IP SGi tunnel (see [3GPP-TS_23.401])
2. via Service Capability Exposure Function (SCEF) (see [3GPP-TS_23.682])

When carrying LWM2M over NB-IoT non-IP mode some limitations and considerations apply which are explained in the following sub-sections.

M.2 NIDD via PtP IP SGi tunnel

As specified in [3GPP-TS_23.401], for mobile originated traffic the P-GW is responsible for creating the IP-packets before sending them via the point-to-point tunnel to the AS. For this, the destination IP address and UDP port for PtP tunnelling based on UDP/IP need to be pre-configured on the P-GW.

As specified in [3GPP-TS_23.401], for mobile terminated traffic, in case PtP tunnelling based on UDP/IP is used, the AS sends the data using UDP/IP encapsulation with the IP address of the UE and the 3GPP defined port for “Non-IP” data. The IP-address of the UE is assigned by the P-GW, however, the UE is not aware of its IP address. The P-GW removes the UDP/IP headers and the data is forwarded via the mobile network to the UE.

[3GPP-TS_29.061] provides further information on how to carry NIDD via the SGi interface.

From the above it can be seen that one limitation for the non-IP transport is the need to pre-configure the destination address in the P-GW so that payloads are correctly relayed. Thus, it is not possible to for a LWM2M client to selectively address more than one LWM2M server via its IP address. As a consequence, the LWM2M client cannot apply separate IP addresses for communicating with

1. different LWM2M Servers
2. separate LWM2M Server and separate LWM2M Bootstrap Server.

Client/Server initiated bootstrap could still be applied, however, for this the LWM2M Server and LWM2M Bootstrap Server would need to be combined and have the same server URI. From a security perspective it is advisable to keep LWM2M Server and LWM2M Bootstrap Server separate. Using factory bootstrap or smartcard bootstrap mode would remove the need of a LWM2M Bootstrap Server.

Another alternative would be to have an intermediary node between SGi and LWM2M server. Such a node could then inspect the CoAP messages for routing information such as URI-host and forward the messages accordingly to different LWM2M servers.

Further mechanisms to remove the above addressing limitation are for further consideration. E.g. a link layer protocol on top of NAS could re-establish addressing capabilities.

Given the required pre-configuration of the destination IP address in the mobile network an IoT platform provider needs to contact the mobile operator to get device-platform connectivity pre-configured (IP address, port number, dedicated APN if desired).

M.3 NIDD via SCEF

[3GPP-TS_23.682] specifies NIDD via SCEF.

[3GPP-TS_23.682] also gives some guidelines how an AS can retrieve small data via the SCEF and suggests the message types “NIDD configuration request/response”, “NIDD submit request/response”, and “NIDD request/response” (see [3GPP-TS_23.682], clause 5.13). However, the definition of required APIs is considered out of 3GPP’s scope.

*Editor’s Note: If OMA provides the required APIs a reference should be added here The APIs should provide at least the following functionality:

- NIDD from SCEF to LWM2M server and vice-versa, including the message types NIDD configuration request/response, NIDD submit request/response, and NIDD request/response
- Information about device awake/sleep transitions
- SCEF-LWM2M server bulk interface might be needed for which OMA would need to define a message structure.

Obviously in non-IP mode the device is not able to address LWM2M server(s) via their IP address. If NIDD via SCEF is selected all data goes via the SCEF. Thus, the UE can only talk to one LWM2M server since there is no additional information available at the IP layer that allows to selectively address more than one LWM2M server. This information would for regular LWM2M be available in the IP header.

An alternative would be to have an intermediary node between SCEF and LWM2M server. Such a node could then inspect the CoAP messages for routing information such as URI-host and forward the messages accordingly to different LWM2M servers.

The LWM2M server needs to identify the UE towards the SCEF via its MSISDN or External Identifier (see [3GPP-TS_23.682]).

M.4 NAS Transport

[3GPP-TS_24.301] defines the transport of user data via the control plane procedure. Two dedicated NAS messages are specified for transferring small data via the MME, see CONTROL PLANE SERVICE REQUEST message and ESM DATA TRANSPORT message in 3GPP [3GPP-TS_24.301]. For initiation of user data transport via the control plane the CONTROL PLAN SERVICE REQUEST message is used which may include ESM DATA TRANSPORT message in its IE “ESM message container”. After the initiation of user data transport via control plane the separate ESM DATA TRANSPORT messages may be used for further transport of user data.

CoAP messages are placed into the IE “User data container” of the ESM DATA TRANSPORT message.

In case DTLS is used the same applies to the DTLS messages.

The user data container has a variable length and the maximum payload size is 32768 bytes. According to 3GPP TS 23.060 the network shall use a maximum packet size of at least 128 octets (this applies to both uplink and downlink). The maximum uplink packet size that the MS shall use can be provided by the network as a part of the session management configuration via the Protocol Configurations Options (PCO) (see [3GPP-TS_24.008] and [3GPP-TS_27.060]). According to [3GPP-TS_24.008] the maximum size for Non-IP link MTU is 1358 octets to prevent fragmentation in the backbone network. The maximum uplink packet size as indicated in the PCO may be retrieved by the LWM2M Server via the Communications Characteristics Object.

Editor’s note: Reference to Communications Characteristics Object to be added.

It has to be noted that there is no segmentation mechanism available for NAS transport. Thus, the LWM2M Client must not exceed the maximum uplink packet size as indicated via the PCO.

Furthermore, the LWM2M Server must not exceed the maximum downlink packet size supported for NAS transport.

The PCO is also used to convey a rate control instruction to the UE i.e. the maximum number of uplink/downlink messages per a specific time unit (see [3GPP-TS_23.401]). This can lead to a delay of LWM2M message delivery in case the rate is exceeded. The LWM2M server can get awareness of any applied rate control via the Cellular Network Connectivity Object (Serving PLMN Rate Control) and via the APN Connection Profile Object (APN Rate Control).

Serving PLMN Rate Control can make the Lwm2M message blocked from the Lwm2M Server side after the limit of the value crosses. In these scenarios Lwm2M Client SHOULD stop expecting responses and wait to cross the period of

silence enforced by the 3GPP network control on message flow from LwM2M Server to LwM2M Client.

Editor's note: Reference to above Objects to be added, including byte control, Serving PLMN Rate Control

According to [3GPP-TS_24.301] the CONTROL PLANE SERVICE REQUEST message and the ESM DATA TRANSPORT message include an IE "Release assistance indication" to inform the network whether or not a downlink data transmission (e.g. acknowledgement or response) subsequent to the uplink data transmission is expected. For mobile originating LWM2M traffic this indicator SHOULD be set accordingly.

'Release assistance indication' value	Recommended Use
00 (binary)	In case an ongoing transaction is expected after an uplink message. This ensures that the MME doesn't initiate the connection release. Example: LWM2M Client responding to READ, WRITE, etc. operations as the LWM2M Client doesn't know how many commands it will receive from the LWM2M Server.
10 (binary)	In case a single response or acknowledgement is expected after an uplink message. This leads to the MME initiating the connection release after the next downlink data transmission. Example: LWM2M Registration Update
01 (binary)	In case no response or acknowledgement is expected after an uplink message. This leads to the MME initiating the connection release immediately. Example: LWM2M Notification

Table: M.4-1 'Release assistance indication' value and Recommended Use

Note: Data transmission speed for uplink and downlink via NAS is expected to be around 300 bit/s, or more.

M.5 Large data transport with NB-IoT

Even NB-IoT is mainly designed for small data delivery it does not preclude delivery of very infrequent large data (e.g. software update/software patches).

[3GPP-TS_23.401] describes a control/user plane switch which could be used e.g. to switch the device communication from control plane to user plane in case a software update is expected. However, there will be devices which do only support communication via the control plane.

IETF has defined segmentation handling at the CoAP layer for large file transfer e.g. firmware updates. Blockwise transfers in CoAP: <https://datatracker.ietf.org/doc/draft-ietf-core-block/>

CoAP block transfer MAY be used with for carrying CoAP over NB-IoT. This option can be used with IP-mode and non-IP-mode.

Editor's note: The RFC will soon be finalised.

An alternative approach is the use of CoAP over TCP. Obviously, this option works only with the IP-mode.

Editor's note: The RFC will soon be finalised.

M.6 Message buffering

NB-IoT devices are expected to be in a sleeping and power saving mode much or most of the time to enable a battery lifetime of several years. In case a device is in sleeping more, or, not reachable for other reasons downlink messages need to be buffered. 3GPP has defined such buffering operation as “extended buffering” at the SCEF (see [3GPP-TS_23.682]) and the S-GW (see [3GPP-TS_23.401]).

It has to be noted that there is a potential issue with DTLS timeout and CoAP CON retransmission timer if the messages are buffered in the mobile network. The could lead to the buffer being filled up with retransmissions.

Furthermore, in case LWM2M queue mode and network buffering are both applied then this could lead to the message being stored in each buffer resulting in duplicated delivery of the message after the device wakes up.

One way for avoiding this would be to use only LWM2M queue mode for buffering messages while the device is not reachable due to sleeping mode or other reasons. In that case “extended buffering” in the network should be deactivated. According to [3GPP-TS_23.401] “extended buffering” can be de-activated per APN, or per subscriber.

Alternatively, the LWM2M server could be configured to only send messages when the device is awake. Obvious precondition for this would be the LWM2M server being aware of the device state.

Editor's note: The device state info is available at the MME which informs the SCEF when the device has woken up from power saving mode (see 3GPP TS 23.682). A trigger mechanism from the SCEF to AS could be used, refer ENCap-M2M API from OMA ARC for such API availability.

It has to be noted that certain NB-IoT implementations need to support also time critical use cases for NB-IoT e.g. fire alarms.

M.7 NB-IoT transport configuration options

Various configuration options for NB-IoT transport are provided via the ConnMgmt enabler.

Editor's note: Reference to be added

M.8 Timer considerations

M.8.1 Introduction

3GPP Rel-13 LTE NB-IoT is aimed at constrained low power IoT devices which require infrequent small data transfer and have a battery life of ~10 years. To minimise power consumption these devices use certain features such as Power Save Mode and extended Idle Mode DRX (eDRX) which govern how often the device wakes up, stays up and reachable. Effective use of these parameters in conjunction with LWM2M Registration Life Time, min/max notify periods, and – if LWM2M queue mode operation is used – ACK_TIMEOUT will help the device to have synchronised set of activities that could optimise its power consumption by avoiding unnecessary wake ups and transmissions.

M.8.2 3GPP Parameters

Parameter	Range	Purpose/how it's used by the device
-----------	-------	-------------------------------------

PSM Timer, Extended T ₃₄₁₂	10min-992 days ¹	Max interval between periodic TAU if there is no other transmission from the device. During this time the device is considered as unreachable and can thereby shut down/deactivate.
Active Timer, T ₃₃₂₄	2sec-31 min	The time the UE has to stay up and remain reachable after transitioning to idle state in case there is pending data from the NW to send out. At the end of T ₃₃₂₄ UE can shut down and deactivate.
Extended DRX ²	5.12sec-174 min	Extended Idle mode DRX
Higher Priority PLMN Search Timer		Interval between periodic searches for higher priority PLMNs when camped on a visited PLMN, i.e. roaming scenario; based on SIM configuration, EFHPPLMN ([3GPP-TS_31.102], section 4.2.6)
Rate Control	TBD	Determines the number of allowed uplink PDU transmissions per deci hour per APN as well as per serving PLMN

Table: M.8.2-1 3GPP Parameters

*Note 1: Table 10.5.163a in [3GPP-TS_24.008] specifies range N to $31*N$ in increments of one where the units of N can be 2 seconds, 30 seconds, 1 min, 10 min, 1 hour, 10 hour or 320 hours. In the context of NB-IoT units of 1 or 10 hours will probably be used in most scenarios?*

Note 2: Extended DRX and PSM can coexist and be configured together.

PSM Timer (Extended T₃₄₁₂), Active Timer (T₃₃₂₄) and Extended DRX can be requested by the device from the network by inclusion of requested values in Attach or TAU requests. On accepting the device request, the network will provide the device with values for these timers which the device should use.

The LWM2M server can configure the values which the device should request from the network via the xxx Resource of the Object. By observing this resource, the LWM2M also receives the finally applied timer in case the network has not accepted the device's request.

M.8.3 LwM2M Parameters

Parameter	Range	Purpose/how it's used by the device
Registration Life Time		Max. interval between client performing registration updates
Pmin		Min. time in second between sending notifications for a resource if any of the notify conditions are met
Pmax		Max. time in seconds between sending successive notifications for a resource if none of other notify conditions are satisfied
ACK_TIMEOUT		CoAP timer used with LwM2M queue mode operation. The LwM2M Client MUST wait at least ACK_TIMEOUT seconds from the last CoAP message it sent to the LwM2M Server before intentionally going offline

Table: M.8.3-1 LwM2M Parameters

M.8.4 Interactions between parameters

From the two tables above it is clear that how often the device wakes up, transmits data and stays awake, is controlled by a combination of parameters defined by 3GPP and OMA that need to be configured in unison to maximise device power efficiency. For example, in the absence of any service data transmission, the device still has to wake up on a regular basis to: a) update its registration with the LwM2M server to make sure its registration stays valid and b) carry out periodic TAU to meet 3GPP requirements. If parameters are configured such that Registration lifetime $\leq T_{3412}$, then the need to perform TAU will automatically be eliminated because when the registration update is performed by the device, the periodic TAU timer will automatically be reset. However, if $T_{3412} < \text{Registration lifetime}$, then the device either wakes up twice to carry out these procedures separately, or unilaterally decides to update its registration on expiry of T_{3412} which will result in the LwM2M server receiving updates more frequently than it had asked for.

Both Active Timer (T3324) and CoAP MAX_TRANSMIT_WAIT are aimed to achieve the same result, namely keeping the device awake long enough to allow queued messages to be sent to the device. Depending on where the queuing occurs, only one of these timers is actually required and can be meaningfully used. It's also worth noting that these timers do not start at the same time, MAX_TRANSMIT_WAIT is started after last transmission, while T3324 starts later when the device has released the connection and returned to idle state.

From 3GPP perspective eDRX determines how often the device needs to wake up and monitor its paging channel. From LWM2M perspective, a device that is monitoring a sensor at the minimum needs to wake up every Pmax to sample sensor data and transmit its value. It also does not need to wake up any more frequently than Pmin to read the sensor data, assuming no filtering & averaging. Pmin, Pmax and eDRX need to be configured such that a device can schedule its activities to minimise the number of times it needs to wake up and transmit data; this is also contingent on any configured Rate Control value being in line with Pmin & Pmax.

M.8.5 Timer implementation options

Effective use of 3GPP timers in conjunction with LWM2M timers will help the device to have synchronised set of activities that could optimise its power consumption by avoiding unnecessary wake ups and transmissions.

The following recommendations apply:

- After bootstrapping, registration or interaction with the LWM2M server, the device needs to examine above mentioned parameters and negotiate 3GPP parameters with the network as appropriate to maximise the power efficiency.
- One set of possible relationships between the various parameters that could provide efficiency is given below.

The customer solution designer will define the values of Pmin and Pmax, and the Infrastructure solution provider will define the values of Extended T3412, LWM2M Registration lifetime and eDRX.

For efficient interaction between mechanisms, the following relationships can be maintained between values of LWM2M and 3GPP parameters:

- $\text{Extended } T_{3412} > \text{LWM2M Registration lifetime} > P_{\text{max}}$, in the scenarios where only Pmax is configured for simple periodic reporting such as a water meter reading Whenever Pmax expires the device will wake up, read the sensor, and sends service data to the server and can optionally send registration update at the same time. This way it will

only need to wake up once every Pmax cycle.

- $P_{min} < eDRX < P_{max} \leq$ translated value of Rate control (in deci-hours) to time interval. The assumption is P_{min} is configured alongside the setting of thresholds on a resource (greater than, less than, step) which means the device needs to wake up every so often (at a minimum every P_{max}) to sample some sensor input. Therefore, when eDRX is also configured the device can simply wake up every eDRX cycle to both sample its sensor input and monitor its paging. It can then evaluate its sensor data against a threshold and decide whether to transmit any data or not. It also does not make sense for rate control to stop the device meeting P_{max} requirements.
- When rate control is applied the P_{min} and P_{max} setting need to be chosen in a way to avoid notifications in a higher rate than rate control allows.