|  |  |  |  |
| --- | --- | --- | --- |
| Description: oma |  | | |
| Lightweight Machine to Machine Technical Specification  Transport Bindings | | | |
| Approved Version 1\_1 – 15 Sep 2017 | | | |
| **Open Mobile Alliance** | | | |
| OMA-TS-LightweightM2M-V1\_1-Transport-20170915-A | | | |
|  | | | | |  |
|  | | | |
|  | | |  |

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.

© 2017 Open Mobile Alliance All Rights Reserved.  
Used with the permission of the Open Mobile Alliance under the terms set forth above.

Contents

1. Scope 5

2. References 6

2.1 Normative References 6

2.2 Informative References 7

3. Terminology and Conventions 9

3.1 Conventions 9

3.2 Definitions 9

3.3 Abbreviations 9

4. Introduction 10

4.1 Version 1.0 11

7. Security 12

7.1 DTLS-based Security 12

7.1.1 Requirements 12

7.1.2 DTLS Overview 12

7.1.3 Ciphersuites 13

7.1.4 Bootstrapping 13

7.1.5 Endpoint Client Name 14

7.1.6 LwM2M and DTLS Roles 14

7.1.7 Pre-Shared Keys 14

7.1.8 Raw Public Keys 15

7.1.9 X.509 Certificates 15

7.1.10 “NoSec” mode 16

7.1.11 Certificate mode with EST 16

7.2 SMS Channel Security 17

7.2.1 SMS “NoSec” mode 17

7.2.2 SMS Secured mode 18

7. Transport Layer Binding and Encodings 22

8.1 Required Features 22

8.2 URI Identifier & Operation Mapping 22

8.2.1 Firewall/NAT 23

8.2.2 Alternate Path 23

8.2.3 Bootstrap Interface 23

8.2.4 Registration Interface 25

8.2.5 Device Management & Service Enablement Interface 26

8.2.6 Information Reporting Interface 30

8.3 Queue Mode Operation 31

8.4 Update Trigger Mechanism 34

8.5 Response Codes 35

8.6 Transport Bindings 37

8.6.1 UDP Binding 38

8.6.2 SMS Binding 38

Appendix A. Change History (Informative) 39

A.1 Approved Version History 39

Figures

[Figure 1: The overall architecture of the LwM2M Enabler. 11](#_Toc492480592)

[Figure 2: The protocol stack of the LwM2M Enabler. 12](#_Toc492480593)

[Figure 19: Example of Client initiated Bootstrap exchange 27](#_Toc492480594)

[Figure 20: Example of Server initiated Bootstrap exchange 28](#_Toc492480595)

[Figure 21: Example register, update and de-register operation exchanges (shorthand in [CoAP] example style, actual messages using CoAP binary headers) 29](#_Toc492480596)

[Figure 22: Example of Device Management & Service Enablement interface exchanges 32](#_Toc492480597)

[Figure 23: Example of Object Creation and Deletion 33](#_Toc492480598)

[Figure 24: Example of an Information Reporting exchange 34](#_Toc492480599)

[Figure 25: Example of Device Management & Service Enablement interface exchanges for Queue Mode 36](#_Toc492480600)

[Figure 26: Example of an Information Reporting exchange for Queue Mode 37](#_Toc492480601)

[Figure 27: Example of Device Management & Service Enablement interface exchanges for Queue Mode with SMS Registration Update Trigger 38](#_Toc492480602)

Tables

[Table 23: Operation to Method and URI Mapping 26](#_Toc492480603)

[Table 24: Operation to Method and URI Mapping 27](#_Toc492480604)

[Table 25: Operation to Method Mapping 30](#_Toc492480605)

[Table 26: Operation to Method Mapping 33](#_Toc492480606)

[Table 27: Response Codes 39](#_Toc492480607)

# Scope

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

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

# References

## Normative References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [3GPP-TS\_23.003] | | | 3GPP TS 23.003 “Numbering, addressing and identification” | |
| [3GPP-TS\_23.032] | | | 3GPP TS 23.032 “Universal Geographical Area Description (GAD)” | |
| [3GPP-TS\_23.038] | | | 3GPP TS 23.038 “Alphabets and language-specific information” | |
| [3GPP-TS\_23.040] | | | 3GPP TS 23.040 “Technical realization of the Short Message Service (SMS)” | |
| [3GPP-TS\_24.008] | | | 3GPP TS 24.008 “Mobile radio interface Layer 3 specification; Core network protocols; Stage 3” | |
| [3GPP-TS\_25.331] | | | 3GPP TS 25.331 “Radio Resource Control (RRC); Protocol specification” | |
| [3GPP-TS\_31.111] | | | 3GPP TS 31.111 “Universal Subscriber Identity Module (USIM)Application Toolkit (USAT)” | |
| [3GPP-TS\_31.115] | | | 3GPP TS 31.115 “Remote APDU Structure for (U)SIM Toolkit applications” | |
| [CoAP] | | Shelby, Z., Hartke, K., Bormann, C., and B. Frank, “The Constrained Application Protocol (CoAP)”  IETF RFC 7252 – June 2014 | | |
| **[CoAP\_Blockwise]** | | C. Bormann, Z. Shelby, “Block-wise transfers in CoAP”, IETF RFC 7959. | | |
| **[CoAP-EST]** | | S. Kumar, P. van der Stok, “EST based on DTLS secured CoAP (EST-coaps)”, draft-vanderstok-core-coap-est-00, October, 2016 | | |
| **[CoRE\_Interface]** | | Z. Shelby, M. Vial, “CoRE Interfaces”, draft-ietf-core-interfaces-01, Nov 2013 | | |
| **[****ETSI TS 102.221]** | | “Smart Cards; UICC-Terminal interface; Physical and logical characteristics”, (ETSI TS 102 221 release 11), [URL:http://www.etsi.org/](http://www.etsi.org/) | | |
| **[ETSI TS 102.223]** | | “Smart Cards; Card Applications Toolkit (CAT) (Release 11)”  [URL:http://www.etsi.org/](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/](http://www.etsi.org/) | |
| **[FLOAT]** | | IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008 | | |
| **[GLOBALPLATFORM]** | | GlobalPlatform v2.2.1 - January 2011 - | | |
| **[GP SCP03]** | | GlobalPlatform Secure Channel Protocol 03 (SCP 03) Amendment D v1.1 Sept 2009 | | |
| [IEEE 754-2008] | | | IEEE Computer Society (August 29, 2008). IEEE Standard for Floating-Point Arithmetic. IEEE. doi:10.1109/IEEESTD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008 | |
| [IOPPROC] | | | “OMA Interoperability Policy and Process”, Version 1.13, Open Mobile Alliance™, OMA-IOP-Process-V1\_13, <URL:http://www.openmobilealliance.org/> | |
| [LwM2M-AD] | | | “Lightweight Machine to Machine Architecture”, Open Mobile Alliance™, OMA-AD-LightweightM2M-V1\_0, <URL:http://www.openmobilealliance.org/> | |
| [OBSERVE] | | Hartke, K. “Observing Resources in CoAP”, IETF RFC 7641. | | |
| **[****PKCS#15]** | | “PKCS #15 v1.1: Cryptographic Token Information Syntax Standard”, RSA Laboratories, June 6, 2000. URL:<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf> | | |
| [RFC2119] | | | “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, <URL:http://www.ietf.org/rfc/rfc2119.txt> | |
| [RFC2234] | | | “Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November 1997, <URL:http://www.ietf.org/rfc/rfc2234.txt> | |
| [RFC4122] | | “A Universally Unique Identifier (UUID) URN Namespace”, P. Leach, et al. July 2005, <URL:http://www.ietf.org/rfc/rfc4122.txt> | | |
| **[RFC5246]** | | The Transport Layer Security (TLS) Protocol Version 1.2 | | |
| **[RFC5280]** | | D. Cooper, et al., “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC 5280, May 2008. | | |
| **[RFC5289]** | | TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM) | | |
| **[RFC5487]** | | Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode | | |
| [RFC5958] | | S. Turner, “Asymmetric Key Packages”, RFC 5958, August 2010. | | | |
| [RFC6347] | | Rescorla, E. and N. Modadugu, “Datagram Transport Layer Security Version 1.2”, [RFC 6347](http://tools.ietf.org/html/rfc6347), January 2012. | | |
| [RFC6655] | | McGrew, D. and D. Bailey, “AES-CCM Cipher Suites for TLS”, RFC6655, July 2012. | | |
| [RFC6690] | | Shelby, Z. “Constrained RESTful Environments (CoRE) Link Format”, RFC6690, Aug 2012. | | |
| [RFC7292] | | K. Moriarty, et al., “PKCS #12: Personal Information Exchange Syntax v1.1”, RFC 7292, July 2014. | | | |
| [SENML] | | C. Jennings, Z. Shelby, J. Arkko, “Media Types for Sensor Markup Language (SENML)”, draft-jennings-senml-10 (work in progress), April 2013. | | |
| [TR-069] | | Broadband Forum: “TR-069 CPE WAN Management Protocol” Issue: 1 Amendment 5. <URL:http://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf> | | |
| [WAP-WDP] | | Wireless Application Protocol Forum, “Wireless Datagram Protocol”, June 2001. | | |

## Informative References

|  |  |
| --- | --- |
| [3GPP TS 31.116] | 3GPP TS 31.116 (V10.2.0): “Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications (Release 10)” |
| [3GPP2 C.S0078-0] | 3GPP2 C.S0078-0 (V1.0): “Secured packet structure for CDMA Card Application Toolkit (CCAT) applications” |
| [3GPP2 C.S0079-0] | 3GPP2 C.S0079-0 (V1.0) “Remote APDU Structure for CDMA Card Application Toolkit (CCAT) applications” |
| [DMREPPRO] | “OMA Device Management Representation Protocol, Version 1.3”.  Open Mobile Alliance™. OMA-TS-DM\_RepPro-V1\_3. <URL:http://www.openmobilealliance.org> |
| [DYNAMIC LINK] | “Dynamic Resource Linking for Constrained RESTful Environments”, Z.Shelby, Z.Vial, M.Koster, C.Groves, Oct 2016, draft-ietf-core-dynlink-01 |
| [ETSI TS 102 226] | ETSI TS 102 226 (V11.0.0): “Smart cards; Remote APDU structure for UICC based applications (Release 11)” |
| [ISO/IEC18031:2011] | ISO, “ISO/IEC 18031:2011: Information technology -- Security techniques -- Random bit generation”, November 2011, available at <http://www.iso.org/iso/catalogue_detail.htm?csnumber=54945> |
| [OMADICT] | “Dictionary for OMA Specifications”, Open Mobile Alliance™, OMA-ORG-Dictionary-V2\_9, <URL:http://www.openmobilealliance.org/> |
| [OMNA] | “OMNA Lightweight M2M (LwM2M) Object & Resource Registry”, [URL:http://www.openmobilealliance.org/](http://www.openmobilealliance.org/) |
| [RESOURCE DIRECTORY] | “CoRE Resource Directory”, Z.Shelby, M.Koster, C.Bormann, P.Van Der Stok Oct 2016, draft-ietf-core-resource-directory-09 |
| [RFC3986] | T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, RFC 3986, January 2005. |
| [RFC4086] | D. Eastlake, J. Schiller, S. Crocker, “Randomness Requirements for Security”, RFC 4086, June 2005. |
| [RFC6698] | P. Hoffman, J. Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA”, RFC 6698, August 2012. |
| [RFC7459] | “Representation of Uncertainty and Confidence in the Presence Information Data Format Location Object (PIDF-LO)”, M. Thomson, J. Winterbootom, February 2015. <URL:https://tools.ietf.org/html/rfc7459> |
| [SMS-DTLS] | “Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things”, H. Tschofenig, T. Fossati, July 2016, [URL:http://www.ietf.org/rfc/rfc7925.txt](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> |

# Terminology and Conventions

## 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.

## Definitions

|  |  |
| --- | --- |
| LwM2M Bootstrap-Server Account | LwM2M Security Object Instance with Bootstrap-Server Resource true |
| LwM2M Server Account | LwM2M Security Object Instance with Bootstrap-Server Resource false and associated LwM2M Server Object Instance |
| Kindly consult [OMADICT] for more definitions used in this document. | |

## Abbreviations

# Introduction

This enabler defines the application layer communication protocol between a LwM2M Server and a LwM2M Client, which is located in a LwM2M Device. The OMA Lightweight M2M enabler includes device management and service enablement for LwM2M Devices. The target LwM2M Devices for this enabler are mainly resource constrained devices. Therefore, this enabler makes use of a light and compact protocol as well as an efficient resource data model.

A Client-Server architecture is introduced for the LwM2M Enabler, where the LwM2M Device acts as a LwM2M Client and the M2M service, platform or application acts as the LwM2M Server. The LwM2M Enabler has two components, LwM2M Server and LwM2M Client. Four interfaces are designed between these two components as shown below:

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

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



Figure 1: The overall architecture of the LwM2M Enabler.



Figure 2: The protocol stack of the LwM2M Enabler.

## Version 1.0

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

0. Security Object

1. Server Object

2. Access Control Object

3. Device Object

4. Connectivity Monitoring Object

5. Firmware Update Object

6. Location Object

7. Connectivity Statistics Object

# Security

The LwM2M protocol is based on [CoAP] principles and utilizes the UDP and SMS transport channel bindings of the protocol. The LwM2M protocol utilizes DTLS with these channel bindings to implement authentication, confidentiality, and data integrity features of the protocol between communicating LwM2M entities. As an alternative, lower layer security may be used, as described in Section 7.2.

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

LwM2M supports three different types of credentials, namely

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

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

The LwM2M protocol specifies that authorization of LwM2M Servers to access Object Instances and Resources within the LwM2M Client is provided through Access Control Object Instances within the LwM2M Client.

## DTLS-based Security

### Requirements

For authentication of communicating LwM2M entities, the LwM2M protocol requires that all communication between LwM2M Clients and LwM2M Servers as well as LwM2M Clients and LwM2M Bootstrap-Servers are authenticated using mutual authentication. This means that a:

* LwM2M Client MUST authenticate a LwM2M Server prior to exchange of any information.
* LwM2M Server MUST authenticate a LwM2M Client prior to exchange of any information.
* LwM2M Client MUST authenticate a LwM2M Bootstrap-Server prior to exchange of any information.
* LwM2M Bootstrap-Server MUST authenticate a LwM2M Client prior to exchange of any information.

For confidentiality and data integrity of information between communicating LwM2M entities, the LwM2M protocol requires that all communication between LwM2M Clients and LwM2M Servers as well as LwM2M Clients and LwM2M Bootstrap-Servers are encrypted and integrity protected. This means that a:

* LwM2M Client MUST encrypt and integrity protect data communicated to a LwM2M Server.
* LwM2M Server MUST encrypt and integrity protect data communicated to a LwM2M Client.
* LwM2M Client MUST encrypt and integrity protect data communicated to a LwM2M Bootstrap-Server.
* LwM2M Bootstrap-Server MUST encrypt and integrity protect data communicated to a LwM2M Client.

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

The use of DTLS fulfils these requirements.

### DTLS Overview

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

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

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

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

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

### Ciphersuites

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

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

* + 1. Prior to TLS 1.1 IV selection is broken. The solution is to use TLS 1.1 or higher, and there is a work-around for earlier version using record splitting. Since this specification relies on DTLS 1.2 this concern is not applicable.
    2. Implementing authenticated decryption (checking padding and mac) without any side channel is hard (see Lucky 13 attack and its variants). The solution is to use the encrypt-then-mac extension defined in RFC 7366, which is recommended.

### Bootstrapping

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

1. for providing UDP channel security in “Client Registration”, “Device Management & Service Enablement”, and “Information Reporting” Interfaces if the LwM2M Security Object Instance relates to a LwM2M Server, or,
2. for providing channel security in Bootstrap Interface if the LwM2M Security Object instance relates to a LwM2M Bootstrap-Server.
3. for protecting the communication with a firmware repository server when the LwM2M Client receives a URI in the Package URI of the Firmware Update object.

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

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

LwM2M Clients MUST either be provisioned for use with a LwM2M Server (manufacturer pre-configuration bootstrap mode) or else be provisioned for use with an LwM2M Bootstrap-Server. Any LwM2M Client, which supports client or server initiated bootstrap mode, MUST support at least one of the following secure methods:

1. Bootstrapping with a strong (high-entropy) pre-shared secret, as described in Section 7.1.7. The ciphersuites defined in Section 7.1.7 MUST NOT be used with a low-entropy secret or with a password.
2. Bootstrapping with a raw public key or certificate-based method (as described in Section 7.1.8 and Section 7.1.9).

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

NOTE: The above security methods can also be used by the LwM2M Bootstrap-Server to provision KIc and KID for the SMS Secured Packet Structure mode (see Section 7.2.2 for SMS Secured Packet Structure mode).

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

### Endpoint Client Name

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

### LwM2M and DTLS Roles

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

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

1. Value in the Server Name Indication (SNI) extension used in the DTLS exchange,
2. Endpoint Client Name,
3. Identifier used with the credential, such as the identifier contained in the DTLS server certificate, and
4. Value in the LwM2M Server URI Resource.

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

### Pre-Shared Keys

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

* TLS\_PSK\_WITH\_AES\_128\_CCM\_8, as defined in [RFC6655]
* TLS\_PSK\_WITH\_AES\_128\_CBC\_SHA256, as defined in [RFC5487]

A LwM2M Client MUST support the Pre-Shared Key mode of DTLS with at least one of the ciphersuites specified for the LwM2M Server.

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

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

### Raw Public Keys

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

* TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8, as defined in [RFC6655]
* TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256, as defined in [RFC5289]

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

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

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

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

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

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

### X.509 Certificates

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

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

* TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8, as defined in [RFC7251].
* TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256, as defined in [RFC5289]

If a LwM2M Client supports X.509 Certificate mode it MUST support at least one of the ciphersuites supported by the LwM2M Server.

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

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

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

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

The algorithm for verifying the service identity, as described in [RFC](https://tools.ietf.org/html/rfc6125) [6125](https://tools.ietf.org/html/rfc6125) [[RFC6125](https://tools.ietf.org/html/rfc6125)], is essential for ensuring proper security when certificates are used and MUST be implemented and used by the DTLS client. Terms like reference identifier and presented identifier are defined in RFC 6125.

Comparing the reference identifier against the presented identifier obtained from the certificate is required to ensure the DTLS client is communicating with the intended DTLS server. Since only the domain-issued certificate mode is supported by this specification the DTLS client compares the certificate from the Server Public Key Resource with the certificate provided in the DTLS handshake additionally comparing the reference identifier against the presented identifier is step for future-proofing in the anticipation of supporting other PKIX validation modes. Similarly, a DTLS client running on a LwM2M Server would need to obtain the certificate of the DTLS server running on the LwM2M Client from some repository.

The algorithm description from RFC 6125 assumes that fully qualified DNS domain names are used. If a server node is provisioned with a fully qualified DNS domain, then the DTLS server certificate MUST contain the fully qualified DNS domain name or "FQDN" as dNSName [RFC5280]. For CoAP, the coaps URI scheme is described in Section 6.2 of [RFC7252]. This FQDN is stored in the SubjectAltName or in the leftmost Common Name (CN) component of the subject name, as explained in Section 9.1.3.3 of [RFC7252], and used by the DTLS client to match it against the FQDN used during the lookup process, as described in [RFC6125].

Note that the Server Name Indication (SNI) extension [[RFC6066](https://tools.ietf.org/html/rfc6066)] allows a DTLS client to tell a DTLS server the name of the DTLS server it is contacting. This is an important feature when the server is part of a hosting solution where multiple virtual servers are using a single underlying network address. [Section 3 of [RFC6066]](https://tools.ietf.org/html/rfc6066#section-3) only allows FQDN hostname of the DTLS server in the ServerName field. For the DTLS client running on a LwM2M Server the SNI extension allows the LwM2M Server to indicate what certificate it is expecting.

In some deployment scenarios DNS is not used and hence LwM2M Clients need to follow a different procedure.

If the CoAP URI stored in the "LwM2M Server URI" Resource contains an IP literal, such as coaps://[2001:db8::2:1]/, then certificate provided by the server also has to contain such an IP address in the Common Name (CN) component of the server certificate or in a field of URI type in the SubjectAltName set.

The procedure for a client using such certificates is as follows:

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

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

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

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

### “NoSec” mode

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

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

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

### Certificate mode with EST

This mode uses the configuration of the certificate mode defined in Section 7.1.9 with the following changes; instead of generating the certificate and the private key for the client by the LwM2M Bootstrap Server and to provision it to the LwM2M Client the Bootstrap Server MUST set the “Security Mode” Resource to value 4 and provisions the certificate of the DTLS server to the “Server Public Key” Resource. This triggers the LwM2M Client to locally generate a public / private key pair on the LwM2M Client and to initiate an EST over CoAP protocol exchange [CoAP-EST] to obtain a certificate. The EST over CoAP specification [CoAP-EST] profiles the use of EST for use in constrained environments.

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

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

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

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

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

but only the first two are mandatory to implement.

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

## SMS Channel Security

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

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

LwM2M Clients supporting SMS, when the SMS Channel is only used for debugging purposes MAY support the NoSec mode.

LwM2M Clients supporting UDP and SMS, when the SMS Channel is only used for triggering as defined in chapter 8.4 MUST support the adequate mechanism for securing UDP Channel as defined in chapter 7.1 UDP channel security. Those clients MAY use any SMS security mode. In particular SMS NoSec mode can be used for SMS triggering since all other communication will be secured by UDP channel security.

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

LwM2M Clients supporting SMS for communications other than triggering, or supporting only the SMS Channel MUST support SMS Secured Mode. In any security mode except for debugging purposes, when an SMS message is received from an MSISDN that is not recorded in the LwM2M Server SMS Number resource of the LwM2M Server Access Security, the SMS message MUST be silently ignored.

### SMS “NoSec” mode

It is highly recommended to always use LwM2M with one of the security mechanisms described in this section. However, there are few scenarios and use cases where security is provided by lower layers. For example, LwM2M devices in a controlled environment behind a gateway, or, tests focussing first on other functions before performing end-to-end tests including security.

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

### SMS Secured mode

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

A LwM2M Client which uses the SMS binding MUST either be directly provisioned for use with a target LwM2M Server (Factory Bootstrap or Bootstrap from Smartcard) or else be able to bootstrap via the UDP binding.

The end-point for the SMS channel (delivery of mobile terminated SMS, and sending of mobile originated SMS) MAYbe either on the Smartcard or on the Device. When the LwM2M Client device doesn’t support a Smartcard, the end-point is on the LwM2M Client device.

A LwM2M Client, Server or Bootstrap-Server supporting SMS binding MUST discard SMS messages which are not correctly protected using the expected parameters stored in the “SMS Binding Key Parameters” Resource and the expected keys stored in the “SMS Binding Secret Keys” Resource, and MUST NOT respond with an error message secured using the correct parameters and keys.

#### Device end-point

The Secured Packet Structure is based on [3GPP TS 31 115] / [ETSI TS 102 225] which was originally designed for securing packet structures for UICC based applications. However, for LwM2M it is suitable for securing the SMS payload exchanged between client and server. Usage of Secured Packet Structure Packet mode in LwM2M device needs evolution towards the introduction of a secure environment. The intention is to evolve the specifications in the next LwM2M release.

In LwM2M Enabler 1.0 if the SMS channel end-point is on the Device, the Channel security for [CoAP] is based on the Datagram Transport Layer Security (DTLS) [RFC6347]. For that reason the main lines of section 7.1 on “DTLS-based Security” relative to DTLS binding on CoAP are also applicable to that section.

The following sub-sections describes how to bind CoAP/DTLS message to the SMS channel and specifies the restrictions on DTLS for fitting the SMS channel specific functioning and narrow bandwidth. The text has been re-used from Appendix A of [SMS-DTLS].

##### DTLS Handshake considerations

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

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

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

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

##### DTLS Message Segmentation and Re-Assembly Consideration

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

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

##### Multiplexing Security Association

This functionality specified in [SMS-DTLS] could authorize to address multiple LwM2M Clients in the same devices, each Clients having a specific identifier, carried by an extra header (7bytes) based on WAP User Datagram Protocol specification [WAP-WDP]. This functionality would require to subtract additional 7 bytes (WDP header) from the SMS effective payload and is not supported in LwM2M release 1.0. Later version of OMA LwM2M may support it through a new SMS DTLS mode (DTLS mode with support to Multiplexing Security Associations), and managing a header of 7 bytes in addition to the one specified in section 7.2.2.1.6.

##### DTLS supported authentication modes considerations

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

##### Timers values for DTLS

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

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

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

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

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

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

##### Header Definitions (for one SMS)

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TPDU (140 bytes) | | | | |
| DTLS (29 bytes) | | | CoAP + Effective Payload | |
| Header (13) | Nonce (8) | ICV (8) |  |  |
|  | | | CoAP ( 4 bytes) | Effective Payload ( 107 bytes) |

Model calculation using these header definitions,

* Overall TPDU : 140 bytes
  + DTLS requires 29 bytes: 13 bytes header according to (RFC 6347 and Appendix B of [RFC7925] + 8 bytes for the explicit nonce and 8 bytes for the integrity check value when an AES-128-CCM-8 ciphersuite is used. This ciphersuite uses a short integrity check value.
  + CoAP header of variable length with at least 4 bytes [CoAP]
  + Available bytes for the effective LwM2M payload from one SMS: 107 bytes

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

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

Model calculation using these header definitions,

* + DTLS takes 29 bytes: 13 bytes (reference, RFC 6347) of header + 16 bytes of integrity check for CoAP in DTLS [RFC 6655] . Cipher suite mandated by CoAP (AES-128)
  + CoAP header 4+8 [CoAP] (Token field required)
  + Available bytes for the effective LwM2M Payload from one SMS: 99 bytes

#### Smartcard end-point

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

The following settings MUST be applied:

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

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

##### Secure SMS Transfer to UICC

A SMS Secured Packet encapsulating a CoAP request received by the LwM2M device, MUST be – according to [ETSI TS 102 225]/[3GPP TS 31.115] - addressed to the LwM2M UICC Application in the Smartcard where it will be decrypted, aggregated if needed, and checked for integrity.

If decryption and integrity verification succeed, the message contained in the SMS MUST be provided to the LwM2M Client.

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

The mechanism for providing the decrypted CoAP Request to the LwM2M Client relies on basic GET\_DATA commands of [GP SCP03] .This data MUST follow the format as below:

data\_rcv \_ ::= <address> <coap\_msg>

address ::= TP\_OA ; originated address

coap\_msg ::= CoAP\_TAG <coap\_request\_length> <coap\_request>

coap\_request\_length ::= 16BITS\_VALUE

coap\_request ::= CoAP message payload

NOTE: In current LwM2M release, the way the LwM2M Client Application is triggered for retrieving the available message from the Smartcard is device specific: i.e. a middle class LwM2M Device implementing [ETSI TS 102 223] ToolKit with class “e” and “k” support could be automatically triggered by Toolkit mechanisms, whereas a simpler LwM2M device could rely on a polling mechanisms on Smartcard for fetching data when available.

##### Secured SMS Transfer to LwM2M Server

For sending a CoAP message to the LwM2M Server, the LwM2M Client prepares a data containing the right TP-DA to use, concatenated with the CoAP message and MUST provide that data to the LwM2M UICC Application in using the [GP SCP03] STORE-DATA command.

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

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

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

#### SMS Secured Packet Binding for CoAP messages

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

* The “Command Packet” command specified in [3GPP 31.115] /[ETSI TS 102 225] MUST be used for both CoAP Request and Response message
* The Structure of the Command Packet contained in the Short Message MUST follow [3GPP 31.115] specification
* SPI MUST be set as follow (see coding of SPI in [ETSI TS 102 225] section 5.2.1):
  + use of cryptographic checksum
  + use of ciphering
    - The ciphering and crypto graphic checksum MUST use either AES or Triple DES
    - Single DES MUST NOT be used
    - AES SHOULD be used
    - When Triple DES is used , then it MUST be used in outer CBC mode and 3 different keys MUST be used
    - When AES is used it MUST be used with CBC mode for ciphering (see coding of KIc in [ETSI TS 102 225] section 5.2.2) and in CMAC mode for integrity (see coding of KID in [ETSI TS 102 225] section 5.2.3).
  + process if and only if counter value is higher than the value in the RE
  + PoR depends on LwM2M Server Policy
* TAR MUST be set to ‘B2 02 03’ value for the LwM2M UICC Application as registered in [ETSI TS 101 220] Appendix D
* Secured Data : contains the Secured Application Message which MUST be coded as a BER-TLV, the Tag (TBD : e.g 0x05) will indicate the type (e.g CoAP type) of that message

# Transport Layer Binding and Encodings

The LwM2M interfaces use the IETF Constrained Application Protocol [CoAP] as an underlying transfer protocol across IP and SMS bearers. This protocol defines the message header, request/response codes, message options and retransmission mechanisms. This section defines the subset of features from the IETF CoAP specification to be used by LwM2M interfaces.

## Required Features

For realization of the LwM2M interfaces, only the basic binary CoAP message header, and a small subset of options are required. This section explicitly defines the features of the CoAP standard that are required for LwM2M.

* The 4-byte binary CoAP message header is defined in Section 3 of [CoAP]. This same base message is used for Request and Response interactions.
* Confirmable, Acknowledgement and Reset messages MUST be supported. The Reset message is used as a message layer error message in response to a malformed Confirmable message. Non-Confirmable messages MAY be used by a Client for sending Information Reporting notifications as per [Observe].
* GET, PUT, POST and DELETE methods MUST be supported. LwM2M Operations map to these methods.
* A subset of Response Codes MUST be supported for LwM2M response message mapping.
* The Uri-Path Option MUST be supported to indicate the identifier of the interface, Object Instance and Resource being requested.
* The Location-Path Option MUST be supported to indicate the handle of a registration for future update and delete operations.
* The Uri-Query Option MUST be supported.
* The Content-Format Option MUST be used to indicate the data format of the payload.
* The Accept Option MAY be included in a LwM2M Server data request, to specify the payload Content-Format this Server prefers to receive. The Client returns the preferred Content-Format if available. If this Accept option is not given or if the LwM2M Client doesn’t support that option, the LwM2M Client will use its own preferred data format reported in the Content-Format of the response message. If the preferred Content-Format cannot be returned, then a 4.06 “Not Acceptable” value MUST be sent as a response.
* The Token Option MAY be used to enable multiple requests in parallel with an endpoint, and MUST be supported for the Information Reporting interface.
* CoAP Blockwise transfer for CoAP MUST be supported by the LwM2M Client when the Firmware Update Object (ID:5) is implemented by the client and MUST be supported by the LwM2M Server.  
  This functionality is motivated by limitations of CoAP, as defined in RFC 7252 [CoAP] since CoAP was not designed for transmission of large payloads. Because the CoAP header itself does not contain length information the UDP length header is used instead. The maximum UDP datagram size is limited to ~64 KiB and transmitting data beyond the (path) maximum transmission (MTU) size will additionally lead to inefficiency because of fragmentation at lower layers (IP layer, adaptation layer, and link layer). Blockwise Transfer for CoAP [draft-ietf-core-block-20] was specifically designed to lift this limitation in order to transfer large payloads larger than ~64 KiB via CoAP, such as firmware images.  
  Note: [CoAP\_Blockwise] is also beneficial for use with firmware images smaller than 64 KiB since the block-wise transmission allows the server to deliver firmware images in chunks suitable to the MTU and thereby avoiding fragmentation at lower layers. A LwM2M client may choose to support block-wise transfer for objects other than the Firmware Update object. This may, for example, be useful with objects that are larger in size, such as the security object which may contain certificates. The specifics of how this functionality is utilized by a LwM2M Server are out of scope for this release of LwM2M.

## URI Identifier & Operation Mapping

Although CoAP supports a URI in requests, it is not used in the same way as in HTTP. The URI in CoAP is broken down into binary parts, minimizing overhead and complexity. In LwM2M only path segment and query string URI components are needed. The URI path is used to simply identify the interface, Object Instance or Resource that the request is for, and is encoded in Uri-Path options. The LwM2M Registration interface also makes use of query string parameters to pass on meta-data with the request separately from the payload. Each query parameter is encoded in a Uri-Query Option. Likewise, the LwM2M operations for each interface are mapped to CoAP Methods. All the LwM2M operations except “Notify” MUST be Confirmable CoAP message and “Notify” can be either Confirmable or Non-Confirmable CoAP message when UDP Transport Layer is used.

### Firewall/NAT

For a firewall to support LwM2M, it should be configured to allow outgoing UDP packets to destination port 5683 (other ports can be configured), and allow incoming UDP packets back to the source address/port of the outgoing UDP packet for a period of at least 240 seconds. These UDP packets may contain DTLS or CoAP payloads. When a firewall is configured as such any LwM2M Clients behind it should use Queue Mode.

For a firewall to support LwM2M it can be configured to allow both outgoing and incoming UDP packets to destination port 5683 (other ports can be configured). These UDP packets may contain DTLS or CoAP payloads. When a firewall is configured as such any LwM2M Clients behind it are not required to use Queued Mode, but may use it for other reasons (e.g., a battery powered sleeping device).

Any LwM2M Clients behind a NAT can use Queued Mode. There are other mechanisms to transverse a NAT, however they are out of scope for the LwM2M Enabler.

### Alternate Path

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

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

This link MUST NOT contain numerical URI segment.

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

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

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

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

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

### Bootstrap Interface

The bootstrap interface is used to optionally configure a LwM2M Client so that it can successfully register with a LwM2M Server. The client bootstrap operation is performed by sending a CoAP POST request to the LwM2M Bootstrap-Server at the /bs path including the Endpoint Client Name as a query string parameter. When bootstrap operation is terminated the Bootstrap-Server MUST send a Bootstrap-Finish indication.

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

In client initiated bootstrap, when the Bootstrap-Server receives Bootstrap-Request operation, or in server initiated bootstrap, the Bootstrap-Server performs Write, Discover and/or Delete operations. The Delete operation targets an Object or an Object Instance, the Discover operation targets an Object, while a Write operation targets Object, Object Instance or a Resource. The Write, Discover and Delete operations can be sent multiple times. Only in Bootstrap Interface, Delete operation MAY target to “/” URI to delete all the existing Object Instances - except LwM2M Bootstrap-Server Account - in the LwM2M Client, for initialization purpose before LwM2M Bootstrap-Server sends Write operation(s) to the LwM2M Client. Different from “Write” operation in Device Management and Service Enablement interface, the LwM2M Client MUST write the value included in the payload regardless of an existence of the targeting Object Instance(s) or Resource and access rights.

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

The Bootstrap-Server MUST send finish indication after it has sent all object instances/resources. Bootstrap-Server send finish message by sending CoAP POST to “/bs” location path with empty payload.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | CoAP Method | URI | Success | Failure |
| **Bootstrap-Request** | POST | /bs?ep={Endpoint Client Name} | 2.04 Changed | 4.00 Bad Request  4.15 Unsupported content format |
| **Write** | PUT | /{Object ID}/{Object Instance ID}/ {Resource ID} | 2.04 Changed | 4.00 Bad Request |
| **Delete** | DELETE | /{Object ID}/{Object Instance ID} | 2.02 Deleted | 4.00 Bad Request |
| **Discover** | GET Accept: application/link-format | /{Object ID} | 2.05 Content | 4.00 Bad Request  4.04 Not Found |
| **Bootstrap-Finish** | POST | /bs | 2.04 Changed | 4.00 Bad Request  4.06 Not Acceptable |

Table 23: Operation to Method and URI Mapping



Figure 19: Example of Client initiated Bootstrap exchange



Figure 20: Example of Server initiated Bootstrap exchange

### Registration Interface

The registration interface is used by a LwM2M Client to register with a LwM2M Server, identified by the LwM2M Server URI.

Registration is performed by sending a CoAP POST to the LwM2M Server URI /rd, with registration parameters passed as query string parameters as per Table 24 and Object and Object Instances included in the payload as specified in Section 5.3.1. The response includes Location-Path Options, which indicate the path to use for updating or deleting the registration. The LwM2M Server MUST return a location under the /rd path segment.

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

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

Registration update is performed by sending a CoAP POST to the Location path returned to the LwM2M Client as a result of a successful registration.

De-registration is performed by sending a CoAP DELETE to the Location path returned to the LwM2M Client as a result of a successful registration.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | CoAP Method | URI | Success | Failure |
| **Register** | POST | /rd?ep={Endpoint Client Name}&lt={Lifetime}&sms={MSISDN} &lwm2m={version}&b={binding} | 2.01 Created | 4.00 Bad Request, 4.03 Forbidden, 4.12 Precondition Failed |
| **Update** | POST | /{location}?lt={Lifetime}&sms={MSISDN} &b={binding} | 2.04 Changed | 4.00 Bad Request, 4.04 Not Found |
| **De-register** | DELETE | /{location} | 2.02 Deleted | 4.00 Bad Request, 4.04 Not Found |

Table 24: Operation to Method and URI Mapping

Note: Throughout the present document the format of the MSISDN must be as specified in [3GPP-TS\_23.003]. According to this definition “+” is not preceding the country code.



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

### Device Management & Service Enablement Interface

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

An Object Instance or Resource is Read by sending a CoAP GET to the corresponding path. The response includes the value in the corresponding Plain Text, Opaque, TLV or JSON format according to the specified Content-Format (see section 6.4).The request MAY specify an Accept option containing the preferred Content-Format to receive. When the specified Content-Format is not supported by the LwM2M Client, the request MUST be rejected.

An Object Instance or Resource is Written to by sending either a CoAP PUT or a CoAP POST to the corresponding path. The request includes the value to be written in the corresponding Plain Text, Opaque, TLV or JSON format according to the Content-Format option which MUST be specified [CoAP]. The Write request MUST be rejected when the specified Content-Format is not supported by the LwM2M Client

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

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

The list of argument can be empty, 2 arguments of the arguments list are separated by a comma. The syntax of the arguments is provided in Section Execute (5.2.4).

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

An Object Instance is Created by sending a CoAP POST to the corresponding path. The request includes the value to be written in the corresponding TLV or JSON format according to the Content-Format option which MUST be specified. The rules governing the creation of Resources in the targeted Object Instance are specified in section 7.3.2.3 (Operation on Object Instance). If Object Instance is not listed at the request, the LwM2M Client MUST assign ID of that Object Instance and send back Object Instance ID with “2.01 Created” to the LwM2M Server when Object Instance is Created.

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

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

<NOTIFICATION> class Attributes MAY be set by a LwM2M Server using the “Write-Attributes” operation by sending a CoAP PUT on the corresponding path, and can be accessed using the “Discover” operation. The Discover operation (uses a CoAP GET on the corresponding path along with the application/link-format Content type, to retrieve a list of Objects, Object Instances, Resources and their attached attributes, from the LwM2M Client (see Section 5.4.2 for more details on DISCOVER command). With the “Write-Attributes” operation one or more Attributes can be written at a time. The values of these Attributes are used by the Information Reporting interface to determine how often Notifications are sent regarding that Resource. A LwM2M Client MAY support a set of these Attributes for each LwM2M Server it is configured for.

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

As example:

1. Write-Attributes /3/0/9?pmin=1 means the Battery Level value will be notified to the Server with a minimum interval of 1sec; this value is set at the Resource level.
2. Write-Attributes /3/0/9?pmin means the Battery Level will be notified to the Server with a minimum value (pmin) given by the default one (resource 2 of Object Server ID=1), or with another value if this Attribute has been set at another level (Object or Object Instance: see section 5.1.1).
3. Write-Attributes /3/0?pmin=10 means that all Resources of Instance 0 of the Object ‘Device (ID:3)’ will be notified to the Server with a minimum interval of 10 sec; this value is set at the Object Instance level.
4. Write-Attributes /3/0/9?gt=45&stp=10 means the Battery Level will be notified to the Server when:
   1. old value is 20 and new value is 35 due to step condition
   2. old value is 45 and new value is 50 due to gt condition
   3. old value is 50 and new value is 40 due to both gt and step conditions
   4. old value is 35 and new value is 20 due to step condition
5. Write-Attributes /3/0/9?lt=20&gt=85&stp=10 means the Battery Level will be notified to the Server when:
   1. old value is 17 and new value is 24 due to lt condition
   2. old value is 75 and new value is 90 due to both gt and step conditions
   3. old value is 50 and new value is 10 due to both lt and step conditions
   4. old value is 87 and new value is 99 due to step condition

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | CoAP Method | Path | Success | Failure |
| Read | GET Accept: Content Format ID (see section 6.4) | /{Object ID}/{Object Instance ID}/{Resource ID} | 2.05 Content | 4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable |
| Discover | GET Accept: application/link-format | /{Object ID}/{Object Instance ID}/{Resource ID} | 2.05 Content | 4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed |
| Write | PUT Content Format: | /{Object ID}/{Object Instance ID}/{Resource ID} | 2.04 Changed  2.31\* Continue | 4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed, 4.15 Unsupported content format  4.08\* Request Entity Incomplete  4.13\* Request entity too large |
| POST Content Format: | /{Object ID}/{Object Instance ID} |
| Write-Attributes | PUT | /{Object ID}/{Object Instance ID}/{Resource ID}?pmin={minimum period}&pmax={maximum period}&gt={greater than}&lt={less than}&st={step} | 2.04 Changed | 4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed |
| Execute | POST | /{Object ID}/{Object Instance ID}/{Resource ID} | 2.04 Changed | 4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed |
| Create | POST Content Format: | /{Object ID} | 2.01 Created | 4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.15 Unsupported content format |
| Delete | DELETE | /{Object ID}/{Object Instance ID} | 2.02 Deleted | 4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed |

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

Table 25: Operation to Method Mapping



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



Figure 23: Example of Object Creation and Deletion

### Information Reporting Interface

Periodic and event-triggered reporting about Resource values from the LwM2M Client to the LwM2M Server is achieved through CoAP Observation [OBSERVE]. This simple mechanism allows the LwM2M Server to send a GET request with Observe option =0 for an Object, Object Instance, or Resource which results in asynchronous notifications whenever that Object Instance changes (periodically or as a result of an event). Token of CoAP layer is used to match the asynchronous notifications with the Observe GET. The LwM2M Server can cancel the “Observe” operation by sending Reset message as the response for Notify message in which the LwM2M Server is not interested any more. When the LwM2M Client receives a Reset in response of a “Notify” operation, the LwM2M Client MUST cancel the Observation regardless if the Notify was sent as a confirmable CoAP message as defined in [OBSERVE] or as a non-confirmable CoAP message. The LwM2M Server can also cancel the “Observe” operation at any moment, on a specified Resource, or specified Object Instance(s), by sending a GET request with Observe option=1. The LwM2M Server may set the Observe attributes of a Resource to affect the behavior its notifications using the ”Write-Attributes” operation (see Section 5.4.4 Write-Attributes).

Please note that this enabler provides two ways for the LwM2M Server to cancel observation:

1. in response to a “Notify” operation for which it is not interested in any more, the LwM2M Server can send a “Reset Message”.
2. at any moment, by sending a GET request with Observe option=1, the LwM2M Server can cancel an “Observe” operation on a specified Resource, or specified Object Instance(s).

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

Table 26: Operation to Method Mapping



Figure 24: Example of an Information Reporting exchange

## Queue Mode Operation

The LwM2M Server MUST support Queue Mode and the LwM2M Client SHOULD support Queue Mode.

When the LwM2M Client has registered with Current Transport Binding and Mode parameter including “Q” (see chapter 5.4), the LwM2M Server does not immediately send downlink requests on the transport used in Queue Mode, but instead waits until the LwM2M Client is online. As such, the Queue Mode offers functionality for a LwM2M Client to inform the LwM2M Server that it may be disconnected for an extended period of time and also when it becomes reachable again. The LwM2M Server uses this information to adjust timers and relay messages from and to the LwM2M Client accordingly.

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

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

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

If message delivery fails, for example, because the message got lost due to network connectivity issues or because the LMW2M Client was sleeping then CoAP re-transmission behaviour at the LwM2M Server will try to retransmit the message. The CoAP stack at the LwM2M Server will resend the message up to a certain number of attempts, as described in Section 4.2 of [CoAP]. If these retransmission attempts fail, the CoAP stack at the LwM2M Server will give up and inform the LwM2M layer. The LwM2M Server has to inform the application about this failed delivery but this API is outside the scope of the LwM2M specification.

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

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

A typical Queue Mode sequence follows the following steps:

1. The LwM2M Client registers to the LwM2M Server and requests the LwM2M Server to run in Queue mode by using the correct Binding value in the registration.
2. The LwM2M Client is recommended to use the CoAP MAX\_TRANSMIT\_WAIT parameter to set a timer for how long it shall stay awake since last sent message to the LwM2M Server. After MAX\_TRANSMIT\_WAIT seconds without any messages from the LwM2M Server, the LwM2M Client enters a sleep mode.
3. At some point in time the LwM2M Client wakes up again and transmits a registration update message. Note: During the time the LwM2M Client has been sleeping the IP address assigned to it may have been released and / or existing NAT bindings may have been released. If this is the case, then the client needs to re-run the DTLS handshake with the LwM2M Server since an IP address and/or port number change will destroy the existing security context. For performance and efficiency reasons it is RECOMMENDED to utilize the DTLS session resumption.
4. When the LwM2M Server receives a message from the Client, it informs determines whether any messages need to be sent to the LwM2M Client, as instructed by the application server.

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



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

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



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

## Update Trigger Mechanism

When the LwM2M Client has registered with Current Transport Binding and Mode parameter with “UQS”, the LwM2M Server MAY make the LwM2M Client come online and register on UDP by executing Registration Update Trigger Resource in Device Object Instance (refer to Appendix E.2). Below is an example flow how to trigger the LwM2M Client in Queue Mode to send Update message to the LwM2M Server regardless of expiration of Lifetime. Post /1/x/8 would bring the LwM2M Client online to talk to the LwM2M server, where “x” represents the right instance pointing to the server.



Figure 27: Example of Device Management & Service Enablement interface exchanges for Queue Mode with SMS Registration Update Trigger

## Response Codes

This section lists available response codes of each operation. The codes are divided into each interface. These are the only valid response codes defined in for the LwM2M Enabler.

|  |  |  |
| --- | --- | --- |
| **Operations** | **Available CoAP Response Codes** | **Reason Phrase** |
| ***Bootstrap Interface*** | | |
| Bootstrap-Request | 2.04 Changed | Bootstrap-Request is completed successfully |
| 4.00 Bad Request | Unknown Endpoint Client Name |
| Write | 2.04 Changed | “Write” operation is completed successfully |
| 4.00 Bad Request | The format of data to be written is different |
| 4.15 Unsupported content format | The specified format is not supported |
| Discover | 2.05 Content | “Discover” operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.04 Not Found | URI of “Discover” operation is not found |
| Delete | 2.02 Deleted | “Delete” operation is completed successfully |
| 4.00 Bad Request | Bad or unknown URI provided |
| Bootstrap-Finish | 2.04 Changed | Bootstrap-Finished is completed successfully |
| 4.00 Bad Request | Bad URI provided |
| 4.06 Not Acceptable | Inconsistent loaded configuration |
| ***Client Registration Interface*** | | |
| Register | 2.01 Created | “Register” operation is completed successfully |
| 4.00 Bad Request | The mandatory parameter is not specified or unknown parameter is specified  Unknown Endpoint Client Name  Endpoint Client Name does not match with CN field of X.509 Certificates |
| 4.03 Forbidden | The Endpoint Client Name registration in the LwM2M Server is not allowed. |
| 4.12 Precondition Failed | Supported LwM2M Versions of the Server and the Client are not compatible |
| Update | 2.04 Changed | “Update” operation is completed successfully |
| 4.00 Bad Request | The mandatory parameter is not specified or unknown parameter is specified |
| 4.04 Not Found | URI of “Update” operation is not found |
| De-register | 2.02 Deleted | “De-register” operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.04 Not Found | URI of “De-register” operation is not found |
| ***Device Management and Service Enablement Interface*** | | |
| Create | 2.01 Created | “Create” operation is completed successfully |
| 4.00 Bad Request | Target (i.e., Object) already exists  Mandatory Resources are not specified  Content Format is not specified |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Create” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for “Create” operation |
| 4.15 Unsupported content format | The specified format is not supported |
| Read | 2.05 Content | “Read” operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Read” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for “Read” operation |
| 4.06 Not Acceptable | None of the preferred Content-Formats can be returned |
| Write | 2.04 Changed | “Write” operation is completed successfully |
| 2.31 Continue |  |
| 4.00 Bad Request | The format of data to be written is different |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of ”Write“ operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for “Write” operation |
| 4.08 Request Entity Incomplete |  |
| 4.13 Request entity too large |  |
| 4.15 Unsupported content format | The specified format is not supported |
| Delete | 2.02 Deleted | “Delete” operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Delete” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for “Delete” operation |
| Execute | 2.04 Changed | “Execute” operation is completed successfully |
| 4.00 Bad Request | The LwM2M Server doesn’t understand the argument in the payload |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Execute” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for “Execute” operation |
| Write-Attributes | 2.04 Changed | “Write-Attributes” operation is completed successfully |
| 4.00 Bad Request | The format of attribute to be written is different |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Write-Attributes” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for Write-Attributes operation |
| Discover | 2.05 Content | “Discover” operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of “Discover” operation is not found |
| 4.05 Method Not Allowed | Target is not allowed for Discover operation |
| ***Information Reporting Interface*** | | |
| Observe  Cancel Observe | 2.05 Content | Operation is completed successfully |
| 4.00 Bad Request | Undetermined error occurred |
| 4.01 Unauthorized | Access Right Permission Denied |
| 4.04 Not Found | URI of Operation is not found or not supported |
| 4.05 Method Not Allowed | Target is not allowed for the Operation |
| 4.06 Not Acceptable | None of the preferred Content-Formats can be returned |
| Notify | 2.05 Content | “Notify” operation is completed successfully |

Table 27: Response Codes

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

## Transport Bindings

The LwM2M Server and the LwM2M Client MUST support UDP binding specified in Section 8.6.1 UDP Binding and the LwM2M Server SHOULD support SMS binding and the LwM2M Client MAY support SMS binding specified in Section 8.6.2 SMS Binding.

### UDP Binding

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

### SMS Binding

CoAP is used over SMS in this transport binding by placing a CoAP message in the SMS payload using 8-bit encoding. SMS concatenation MAY be used for messages larger than 140 characters. CoAP retransmission is disabled for this binding. An LwM2M Client indicates the use of this binding by including a parameter (sms) in its registration to the LwM2M Server including the node’s MSISDN number. The LwM2M Client MAY interact with the server using both UDP and SMS bindings.

1. Change History (Informative)
   1. Approved Version History

| Reference | Date | Description |
| --- | --- | --- |
| OMA-TS-LightweightM2M-V1\_0-20170208-A | 08 Feb 2017 | Status changed to Approved by TP  TP Ref # OMA-TP-2017-0009-INP\_LightweightM2M-V1\_0\_ERP\_for\_Final\_Approval |
| OMA-TS-LightweightM2M-V1\_0\_1-20170704-A | 04 Jul 2017 | Status changed to Approved by TP  TP Ref # OMA-TP-2017-0029R01-INP\_LightweightM2M\_V1.0.1\_ERP\_for\_Final\_Approval\_Notification |
| OMA-TS-LightweightM2M-V1\_1-Transport-20170915-D | 15 Sep 2017 | OMA-TS-LightweightM2M-V1\_1-Transport-20170915-D has been split from 1\_0\_1 Approved version (see above line for reference) for transport specific contents  Deleted sections 5,6, 7.3 and Appendices B to L |