Lightweight Machine to Machine Technical Specification: Transport Bindings

Approved Version: 1.1 - 2018-07-10

Open Mobile Alliance

OMA-TS-LightweightM2M_Transport-V1_1-20180710-A

master: 06 Aug 2018 16:47:00 *rev: 230b17c*

Table of Contents

Table of Figures

Table of Tables

1. Scope

This document specifies the transport bindings of the Lightweight Machine-to-Machine (LwM2M) protocol version 1.1. The split between the LwM2M core [LwM2M-CORE] and the transport binding specification improves readability, allows a cleaner separation between the LwM2M messaging layer and the underlying protocols for conveying these messages, and ultimately better extensibility.

LwM2M version 1.0 supported multiple transport bindings, namely CoAP over UDP, and over SMS. The UDP and SMS transports can be used with or without DTLS. This specification, LwM2M version 1.1, adds support for CoAP over TCP / TLS, and CoAP over Non-IP, namely 3GPP CIoT and LoRaWAN. CoAP over TCP / TLS, as defined in [CoAP_TCP], offers better firewall traversal, as explained in Section 1 of [CoAP_TCP].

This specification also supports the application layer security protocol OSCORE [OSCORE] which enables support for proxy operations and end-to-end security independently of transport layer protocol.

2. References

2.1. Normative References

[3GPP-TS_23.003]	3GPP TS 23.003 "Numbering, addressing and identification"
[CoAP]	Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "The Constrained Application Protocol (CoAP)", RFC 7252, Jun. 2014.
[CoAP_Blockwise]	C. Bormann, Z. Shelby, "Block-wise transfers in CoAP", RFC 7959, Aug. 2016.
[CoAP_TCP]	C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, Feb. 2018.
[CoAP_ERT]	C. Amsuess, J. Mattsson, G. Selander, "Echo and Request-Tag", draft-ietf-core-echo-request-tag-00, Oct. 2017.
[CoAP-EST]	P. van der Stok, et al., "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-00, Feb., 2018
[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/
[GLOBALPLATFORM]	GlobalPlatform v2.2.1 - January 2011 -
[GP SCP03]	GlobalPlatform Secure Channel Protocol 03 (SCP 03) Amendment D v1.1 Sept 2009
[OBSERVE]	Hartke, K. "Observing Resources in CoAP", RFC 7641, Sep. 2015.
[OSCORE]	G. Selander, J. Mattsson, F. Palombini, L. Seitz, "Object Security for Constrained RESTful Environments", draft-ietf-core-object-security-08, Jan. 2018.
[RFC2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Mar. 1997.
[RFC5246]	T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, Aug. 2008.
[RFC5280]	D. Cooper, et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
[RFC5289]	E. Rescorla, "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, Aug. 2008.
[RFC5487]	M. Badra, Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode, Mar. 2009.
[RFC5869]	H. Krawczyk, P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.
[RFC6347]	Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, Jan. 2012.
[RFC6655]	McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for TLS", RFC 6655, Jul. 2012.
[RFC6690]	Shelby, Z. "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, Aug. 2012.
[WAP-WDP]	Wireless Application Protocol Forum, "Wireless Datagram Protocol", June 2001.

[RFC7925]	H. Tschofenig, T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, Jul. 2016.
[RFC8075]	A. Castellani, et al., "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, Feb. 2017.
[RFC8152]	J. Schaad, "CBOR Object Signing and Encryption (COSE)", RFC 8152, Jul. 2017.
[LwM2M-CORE]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core Layer".
[RFC5958]	S. Turner, "Asymmetric Key Packages", RFC 5958, August 2010.
[RFC7366]	P. Gutmann, "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, September 2014.
[RFC8132]	P. van der Stok, C. Bormann, A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, APril 2017.
[SCRRULES]	"SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://member.openmobilealliance.org/ftp/Public_documents/iop/Permanent_documents/OMA-ORG-SCR_Rules_and_Procedures-V1_0-20060919-A.zip
[3GPP-TS_31.115]	3GPP TS 31.115 "Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications"

Table: 2.1.-1 Normative References

2.2. Informative References

[3GPP-TR_21.905]	3GPP TR 21.905 Vocabulary for 3GPP Specifications
[OMADICT]	"Dictionary for OMA Specifications", Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/
[RFC4086]	D. Eastlake, J. Schiller, S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.
[RFC7457]	Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, February 2015.
[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
[LoRaWAN]	The LoRa Alliance, "LoRaWAN 1.1 Specification", available athttps://www.lora-alliance.org/lorawan-for-developers
[RFC6961]	Pettersen, Y. "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, June 2013.

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 set to 'true'.
LwM2M Server Account	LwM2M Security Object Instance with Bootstrap-Server Resource set to '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

```
&lt;/tbody&gt;
```

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

Table: 3.3.-1 Abbreviations

4. Introduction

This specification defines the transport bindings for the LwM2M messaging protocol used between LwM2M Clients, LwM2M Bootstrap Servers and LwM2M Servers. Figure: 4.-1 The Protocol Stack of the LwM2M Enabler shows the relationships between the transport bindings and the messaging protocol. In particular, this specification defines the following transport bindings:

- CoAP over UDP
- CoAP over DTLS (over UDP)
- CoAP over SMS
- CoAP over DTLS over SMS
- CoAP over TCP
- CoAP over TLS (over TCP)
- CoAP over Non-IP (includes 3GPP CIoT and LoRaWAN)

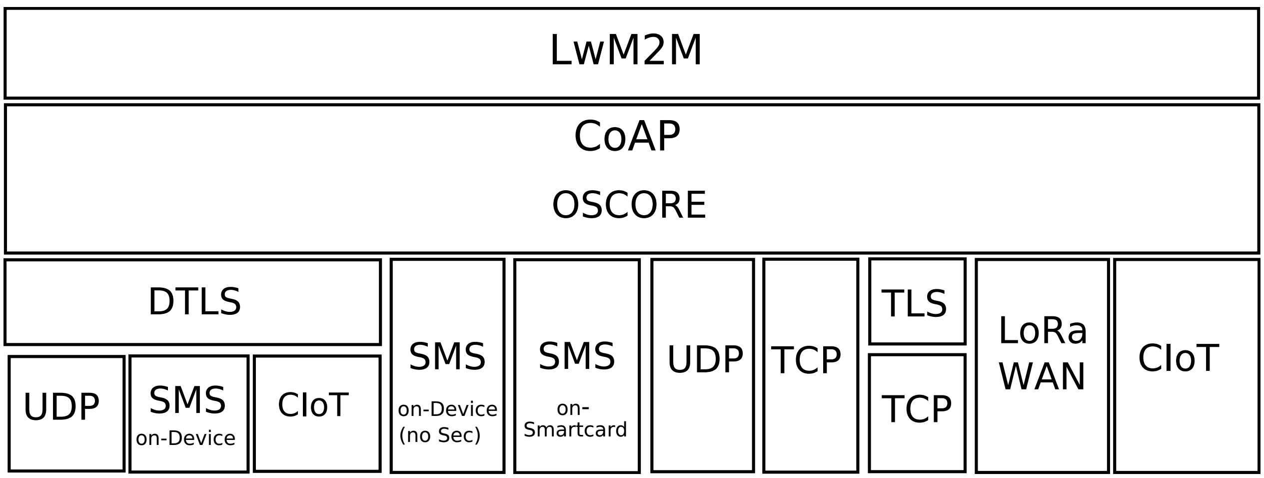

Figure: 4.-1 The Protocol Stack of the LwM2M Enabler

OSCORE can be used with any of the transport bindings including UDP, SMS and TCP, with or without DTLS or TLS.

5. Security

The LwM2M protocol supports various transport bindings and credentials for securely communicating with LwM2M Servers. This configuration information can be provisioned through various forms of bootstrapping methods.

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 specification offers support for all of them. [RFC7925] provides the necessary details about the use of each of these credentials with TLS/DTLS. The LwM2M specification also supports application layer security based on OSCORE with pre-shared secrets. Which credential is best in a given deployment depends on a number of factors, including the threat model and IoT device constraints.

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.

5.1. Security Requirements

LwM2M may under certain circumstances be deployed with security outside the scope of LwM2M TS, as specified in other parts of this section. However, for LwM2M security solutions the requirements specified in this section apply.

The LwM2M protocol requires that all communication between LwM2M Clients, LwM2M Servers and LwM2M Bootstrap-Servers to perform mutual authentication. This means:

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

The LwM2M protocol also requires that all communication between LwM2M Clients and LwM2M Servers as well as between LwM2M Clients and LwM2M Bootstrap-Servers is encrypted and integrity protected. This means:

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

Additionally:

- A security solution MUST be able to provide replay protection of LwM2M Operations
- A security solution MUST be able to securely bind LwM2M responses with LwM2M requests

- For certain operations, the LwM2M Client MUST be able to verify the freshness of the request
- A security solution MUST be able to support secure fragmentation of the messages between LwM2M Server and LwM2M Client into fragments that can be verified separately, in particular in the case of firmware updates.

The security requirements applies also to communication via LwM2M aware and LwM2M unaware intermediate nodes.

In addition to communication security care must also be taken to ensure that data at rest, such as secrets and sensor data, are also secured against unauthorized access.

5.2. TLS/DTLS-based Security

5.2.1. TLS/DTLS Overview

CoAP over UDP [CoAP] is secured using the Datagram Transport Layer Security (DTLS) 1.2 protocol [RFC6347], as defined in Section 9 of [CoAP]. DTLS is a communication security solution for datagram based protocols (such as UDP). More recently support for CoAP over TCP / TLS has been defined in [CoAP-TCP]. TLS 1.2 is specified in [RFC5246]. TLS and DTLS provide mutual authentication, data integrity and confidentiality.

This section provides information related to the use of CoAP over DTLS and CoAP over TLS. Section5.4. SMS Channel Security provides additional information regarding the use of DTLS in an SMS context. Implementations SHOULD be conformant to [RFC7925], which includes not only ciphersuites but also recommendations for the use of TLS/DTLS-specific extensions.

The TLS/DTLS client and the TLS/DTLS server SHOULD keep security state, such as session keys, sequence numbers, and initialization vectors, and other security parameters, established with TLS/DTLS for as long a period as can be safely achieved without risking compromise of 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 TLS/DTLS client and the TLS/DTLS server to secure the communication between the LwM2M Client and the LwM2M Server are obtained using one of the bootstrap modes defined in [LwM2M-CORE].

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. Using different credentials with each LwM2M Server provides better unlinkability properties since each individual LwM2M Server cannot correlate requests made by the LwM2M Client to other LwM2M Servers. Deployment and application specific considerations dictate what approach to use, but using different credentials is generally recommended practice.

5.2.2. Ciphersuites

TLS/DTLS supports the concept of ciphersuites and they are securely negotiated during the handshake. This specification recommends support of a limited number of ciphersuites based on [RFC7925]. These ciphersuites have been chosen because of suitability for IoT devices, security reasons and to improve interoperability. Ciphersuites in TLS/DTLS 1.2 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 the state-of-the-art security requirements.

5.2.3. Elliptic Curves

For ECDHE, the group secp256r1 SHALL be supported. For ECDSA the curve secp256r1 SHALL be supported. Elliptic curve groups of less than 255 bits SHALL not be supported.

5.2.4. Bootstrapping

The Resources in the LwM2M Security Object are used for

1. providing TLS/DTLS communication security for "Client Registration", "Device Management & Service Enablement", and "Information Reporting" Interfaces if the LwM2M Security Object Instance relates to a LwM2M Server, or,

2. providing channel security for the Bootstrap interface if the LwM2M Security Object instance relates to a LwM2M Bootstrap-Server.

3. protecting the communication with a firmware repository 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 depends on the type of credential being used.

When bootstrapping from a 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 a Smartcard with pre-shared secrets, with 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 exists between the LwM2M Server(s) and the SmartCard.

LwM2M Clients MUST either be provisioned for use with a LwM2M Server or else be provisioned for use with a LwM2M Bootstrap-Server. Any LwM2M Client, which supports the client 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 5.2.8.1. Pre-Shared Keys. The ciphersuites defined in Section 5.2.8.1. Pre-Shared Keys MUST NOT be used with a low-entropy secret or with a password.

2. Bootstrapping with a raw public key or a certificate, as described in Section 5.2.8.2. Raw Public Keys and Section 5.2.8.3. X.509 Certificates.

LwM2M Clients MUST be provisioned with credentials that are unique to their device. For full interoperability in all deployment environments, a LwM2M Bootstrap-Server MUST support bootstrapping via pre-shared secrets, raw public keys, and certificates.

NOTE: The LwM2M Bootstrap-Server can also provision KIc and KID for the use of SMS Secured Packet Structure mode (see Section 5.4. SMS Channel Security).

Security credential dynamically provisioned to the LwM2M Client and the LwM2M Server MAY change at any time, even during the lifetime of an ongoing TLS/DTLS session. Since the TLS/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 TLS/DTLS handshake has already been completed. Hence, from a TLS/DTLS protocol point of view such credential change is not automatically recognized and the already established record layer security associations are in use. It is a policy decision for a TLS/DTLS client as well as a TLS/DTLS server implementation to close a connection 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 since TLS/DTLS allows both communication parties to tear down an established TLS/DTLS session for any number of reasons.

The Security Mode Resource in the Security Object determines what credentials are being used by the LwM2M Client and the LwM2M Server or LwM2M Bootstrap-Server, respectively. Currently five security modes are defined, namely

- 0: Pre-Shared Key mode
- 1: Raw Public Key mode
- 2: Certificate mode
- 3: NoSec mode
- 4: Certificate mode with EST

The Enrollment over Secure Transport (EST) bootstrap mode is a certificate management protocol for provisioning certificates from the LwM2M Bootstrap-Server to the LwM2M Client. EST allows the LwM2M Client to generate the key pair locally on the LwM2M Device. The private key therefore never leaves the LwM2M Device. In order to use this mode the "Security Mode" Resource MUST be set to value 4 and the certificate of the TLS/DTLS server MUST be provisioned to the "Server Public Key" Resource. This triggers the LwM2M Client to locally generate a public / private key pair and to initiate an EST over CoAP protocol exchange [CoAP-EST] to obtain a certificate. Once the certificate has been obtained it behaves like the Certificate mode. The EST over CoAP specification [CoAP-EST] profiles the use of EST for use in constrained environments.

Generating high quality random numbers of IoT devices, which is required by the EST mode utilized by this specification, is often a concern. 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 RFC 4086 [RFC4086] or NIST Special Publication 800-90A Revision 1 [SP800-90A].

Compared to the certificate mode with the "Security Mode" Resource set to value 2 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 explained 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. LwM2M Clients SHOULD use an existing key pair or generate the public/private key pair locally since this does not expose the private key to the LwM2M Bootstrap-Server or other parties, i.e., the private key pair SHOULD never leave the device.

Note: The "Secret Key" and the "Public Key or Identity" Resources are not used by the Certificate mode with EST.

Enrollment 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 when the EST over CoAP is used for bootstrapping since the 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 since the LwM2M Bootstrap Server is typically in possession of the required attributes for generating a certificate.

5.2.5. Unbootstrapping

The term 'unbootstrapping' refers to the process of deleting an instance of a LwM2M Security Object. If such an instance of a Security Object is to be deleted related resources and configurations need to be deleted or modified as well. This ensures that there is no orphan data, for example LwM2M Server Object instances with no security context. Therefore, when the Delete operation is sent via the Bootstrap Interface, the Client MUST execute the following procedure.

1. The Server Object Instance, which the deleted Security Object referred to, MUST be deleted. The Client MAY send the "De-register" operation to the LwM2M Server.
2. If there is an Object Instance that can be accessed only by a LwM2M Server of the Server Object Instance (i.e., the LwM2M Server is Access Control Owner and the LwM2M Server can access the Object Instance only in an Access Control Object Instance), the Object Instance and the corresponding Access Control Object Instance MUST be deleted.
3. If an Object Instance can be accessed by multiple LwM2M Servers including the LwM2M Server which Security Object Instance is to be deleted, then

- The ACL Resource Instance for the LwM2M Server in the Access Control Object Instance for the Object Instance MUST be deleted.
- If the LwM2M Server is the Access Control Owner of the Access Control Object Instance, then the Access Control Owner MUST be changed to another LwM2M Server according to the following rules: The Client MUST choose the LwM2M Server who has highest sum of each number assigned to an access right (Write: 1, Delete: 1) for the Access Control Owner. If two or more LwM2M Servers have the same sum, the Client MUST choose one of them as the new Access Control Owner.

4. Observation operations from the LwM2M Server Object Instance MUST be deleted.

Note: To monitor the change of the Access Control Owner the LwM2M Server MAY observe the Access Control Owner Resource.

5.2.6. Endpoint Client Name

This specification recommends, but does not mandate, transmission of the endpoint client name in the Bootstrap-Request and in the Register message. 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 TLS/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.

5.2.7. LwM2M and TLS/DTLS Roles

In LwM2M version 1.1 the LwM2M Client is always the TLS/DTLS client. The LwM2M Bootstrap-Server and the LwM2M Server are always acting as TLS/DTLS servers.

5.2.8. Credential Types

[RFC7925] gives recommendations for three types of credentials, namely pre-shared keys, raw public keys, and X.509

certificates. LwM2M works with all three types of credentials but the performance and security trade-offs for these three mechanisms are different. As a summary, the three credential types have the following properties:

- The pre-shared key profile offers the most efficient solution for integration of TLS/DTLS into LwM2M since pre-shared ciphersuites recommended in [RFC7925] require a minimum amount of flash space as well as RAM size. Symmetric cryptographic algorithms require only a minimal computational overhead. The size of the exchanged messages is also kept at a minimum. There is, however, a downside as well: symmetric keys need to be pre-configured to both communication endpoints.
- The certificate-based profile re-uses widely deployed X.509 certificates. This allows both tools as well as existing infrastructure, such as Certification Authorities (CAs) and the Public Key Infrastructure (PKI), to be re-used. Unlike the typical web browser use of certificates, [RFC7925] specifies the use of certificates for mutual authentication between clients and servers. The use of certificates comes at a price: The use of asymmetric cryptography is more complex to implement, requires more bandwidth for the exchanged messages, is computationally more demanding, and requires a larger code size as well as more RAM. The benefits are, in addition to the re-use of existing technologies, the need to only share the certificates with other communication partners in an authentic fashion and to keep the private key local to each party (at least in the case where EST is used). This property of asymmetric cryptography reduces the risk of exposing private keying material.
- The raw public key profile offers features that sit between the pre-shared key and the certificate-based profile and combines the benefits of these two profiles. The use of asymmetric cryptography offers improved security but avoids the overhead associated with certificates and the PKI.

The following sub-sections define the use of various resources in the LwM2M Security Object for the specific credential types. The "LwM2M Server URI", and the "Bootstrap Server" Resources are populated according to the description in [LwM2M-CORE].

5.2.8.1. Pre-Shared Keys

If a LwM2M Server supports the pre-shared key credentials it MUST support the following:

- TLS_PSK_WITH_AES_128_CCM_8, as defined in [RFC6655] and mandated in [RFC7925]
- TLS_PSK_WITH_AES_128_CBC_SHA256, as defined in [RFC5487].

The LwM2M Client SHOULD NOT use the TLS_PSK_WITH_AES_128_CBC_SHA256 ciphersuite as RFC 7457 [RFC7457] has identified security attacks against these TLS/DTLS ciphersuites.

A LwM2M v1.1 Client MUST support TLS_PSK_WITH_AES_128_CCM_8 and MAY support additional ciphersuites.

This mode requires the following resources of the Security Object 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, described in [RFC7925]. Clients and Servers MUST support arbitrary PSK Identities of up to 128 bytes, as mandated in [RFC7925].
- The "Secret Key" Resource MUST be used to store the PSK, defined in [RFC4279]. Since the default PSK ciphersuite defined in this specification use a 128-bit AES key it is RECOMMENDED to provision a 16 byte (128 bit) key or longer in the Secret Key Resource. Clients and Servers MUST support PSK keys of up to 64 bytes in length, as required by [RFC7925]. Recommendations for generating random keys are provided in RFC 4086 [RFC4086] and in NIST Special Publication 800-90A Revision 1 [SP800-90A].

- The "Server Public Key" Resource MUST NOT be used in the Pre-Shared Key mode.

5.2.8.2. Raw Public Keys

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

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, as defined in [RFC6655] and mandated in [RFC7925]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, as defined in [RFC5289]

The LwM2M Client SHOULD NOT use the TLS_PSK_WITH_AES_128_CBC_SHA256 ciphersuite as RFC 7457 [RFC7457] has identified security attacks against these TLS/DTLS ciphersuites.

A LwM2M v1.1 Client MUST support TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 and MAY support additional ciphersuites. Ciphersuites SHOULD have ECDSA authentication and SHOULD have ECDHE key exchange.

This mode requires the following resources of the Security Object to be populated:

- The "Security Mode" Resource MUST contain the value 1.
- The information stored in the "Public Key or Identity" Resource depends on the "Certificate Usage" Resource, as discussed in Section 5.2.8.7. Certificate Usage Field. If the "Certificate Usage" Resource is missing this Resource MUST be used to store the raw public key encoded using the SubjectPublicKeyInfo structure, as described in [RFC7250].
- The "Secret Key" Resource MUST be used to store the private key of the TLS/DTLS client encoded as OneAsymetricKey, as defined in [RFC5958].
- The "Server Public Key" Resource MUST be used to store the raw public key of the TLS/DTLS server encoded using the SubjectPublicKeyInfo structure, as described in [RFC7250].

This security mode is appropriate for LwM2M deployments where the benefits of asymmetric cryptography are needed but the PKI functionality is undesireable.

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

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

5.2.8.3. X.509 Certificates

The X.509 Certificate mode requires the use of X.509v3 certificates [RFC5280]. The X.509 certificates SHOULD be conformant with the profiling given in [RFC7925]. The curve secp256r1 SHALL be supported. Elliptic curve groups of less than 255 bits SHALL NOT be supported.

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

- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, as defined in [RFC7251] and mandated in [RFC7925]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, as defined in [RFC5289]

The LwM2M Client SHOULD NOT use the TLS_PSK_WITH_AES_128_CBC_SHA256 ciphersuite as RFC 7457 [RFC7457] has identified security attacks against these TLS/DTLS ciphersuites.

A LwM2M v1.1 Client MUST support TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 and MAY support additional ciphersuites. Ciphersuites SHOULD have ECDSA authentication and SHOULD have ECDHE key exchange.

This mode requires the following resources of the Security Object to be populated:

- The "Security Mode" Resource MUST contain the value 2.
- The information stored in the "Public Key or Identity" Resource depends on the Certificate Usage Resource, as discussed in Section 5.2.8.7. Certificate Usage Field. If the Certificate Usage Resource is missing this Resource MUST be used to store the X.509 certificate of the TLS/DTLS client in a DER encoded binary format. This backwards compatibility mode matches the semantic of value 3 ("domain-issued certificate") in the Certificate Usage Resource.
- The "Secret Key" Resource MUST be used to store the private key of the TLS/DTLS client as a DER encoded asymmetric key package OneAsymetricKey version 1 structure, as defined in Section 2 of [RFC5958]. Note that this version of the specification does not support version 2 of RFC 5958 and the encrypted private key info structure defined in Section 3 of [RFC5958].
- The "Server Public Key" Resource MUST be used to store either a trust anchor certificate suitable for path validation of the certificate of the TLS/DTLS server, or directly the certificate of the TLS/DTLS server ("domain-issued certificate mode"). The use of it is explained in more detail below.

When certificate-based authentication is used in TLS/DTLS to protect the LwM2M communication at least two types of certificates need to be distinguished, namely client (or device) certificates and certificates presented by the server. This distinction matters since the identifiers used in the certificates will be different and the procedure for identifier matching will be different as well. The subsequent text talks about the certificates used by the server.

The algorithm for verifying the service identity, as described in Section 4.4.1 of [RFC7925] and in [RFC6125], is essential for ensuring proper security when certificates are used and MUST be implemented and used by the TLS/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 TLS/DTLS client is communicating with the intended TLS/DTLS server. To cater for the case that the Server Public Key Resource directly contains the certificate of the TLS/DTLS server, TLS/DTLS client does not compare the reference identifier against the presented identifier if the certificate from the Server Public Key Resource matches the certificate provided by the TLS/DTLS server during the TLS/DTLS handshake. If that is not the case, the TLS/DTLS client MUST compare the reference identifier against the presented identifier as described below, and perform path validation.

The algorithm description from RFC 6125 assumes that fully qualified DNS domain names (FQDN) are used. If a server node is provisioned with a FQDN, then the TLS/DTLS server certificate MUST contain this "FQDN" (i.e., presented identifier). This FQDN is stored in the SubjectAltName or in the Common Name (CN) component of the subject name, as explained in Section 9.1.3.3 of [RFC7252], and this FQDN is used by the TLS/DTLS client to match it against the FQDN used during the lookup process (i.e., reference identifier) before contacting the LwM2M Server, as described in [RFC6125].

Note that the Server Name Indication (SNI) extension [RFC6066] allows a TLS/DTLS client to tell a TLS/DTLS server the name of the TLS/DTLS server it is trying to contact. This is an important feature when the server is part of a hosting solution where multiple virtual servers are using a single underlying network address. Section 3 of [RFC6066] only allows FQDN hostname of the TLS/DTLS server in the ServerName field. For the TLS/DTLS client running on a LwM2M Server the SNI extension allows the LwM2M Server to indicate what certificate it is expecting. The SNI extension MUST be set by the LwM2M client.

5.2.8.4. Deployments without DNS

In some deployment scenarios, DNS is not used and LwM2M Clients need to use the IP literal, such as coaps://[2001:db8::2:1]/, stored in the "LwM2M Server URI" Resource. To avoid having to use IP addresses also in the Common Name (CN) component of the server certificate or in a field of URI type in the SubjectAltName set this specification uses the Server Name Indication (SNI) Resource.

The procedure for a client is as follows:

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

5.2.8.5. Certificate Expiry

For the use of certificates devices need to be equipped with a real-time clock and need to be provisioned with a reference time since the certificate expiry field needs to be checked.

If the Bootstrap-Server certificate has an expiration time then out-of-band mechanisms may be used to recover from an expired certificate. The LwM2M specification does not offer a way to recover from an expired Bootstrap-Server certificate. Out-of-band mechanism may include replacing the LwM2M Bootstrap-Server certificate via the firmware update mechanism or using a commissioning tool.

The LwM2M Bootstrap-Server or an authorized LwM2M Server MAY use the Current Time Resource of the LwM2M Device Object to provision a date/time reference value to the LwM2M Client. The LwM2M Client SHOULD verify the freshness of the request for setting the Current Time Resource. The LwM2M Client can use this reference value together with the current real-time clock setting to determine the current time to check the expiry date of a LwM2M Server certificate. The ability to modify the value of the Current Time Resource is security critical and therefore appropriate access control settings need to be applied.

5.2.8.6. Certificate Revocation

A LwM2M Client needs to determine when a LwM2M Server certificate has been revoked. Various approaches for dealing with expired LwM2M Server certificates are possible and the most promising candidates are:

1. The LwM2M Client MAY use the error recovery procedures that will be invoked when a TLS / DTLS client encounters a problem verifying the LwM2M Server certificate during the handshake.

2. The LwM2M Client MAY use the Online Certificate Status Protocol (OCSP) as part of the OCSP stapling mechanism [RFC6961]. This does, however, assume that the LwM2M Client and the LwM2M Server regularly execute the TLS / DTLS handshake, which may depend on the deployment. Note that the OCSP stapling comes with an extra cost of conveying the OCSP information inside the TLS / DTLS handshake.

3. The server-initiated bootstrapping procedure is used by a LwM2M infrastructure operator to trigger a LwM2M Client to re-run the bootstrapping exchange to obtain new credentials for use with LwM2M Servers. The server-initiated bootstrapping procedure may not be usable by the LwM2M Server that has gotten its certificate revoked.

If the Bootstrap-Server certificate has been revoked then out-of-band mechanisms may be used to replace the revoked certificate. The LwM2M specification does not offer a way to recover from a revoked Bootstrap-Server certificate. Out-of-band mechanism may include replacing the LwM2M Bootstrap-Server certificate via the firmware update mechanism or using a commissioning tool.

5.2.8.7. Certificate Usage Field

The Certificate Usage Resource in the LwM2M Security Object specifies the semantics 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. RFC 6698 defines for types, which are reused in this specification:

- 0: Certificate usage 0 ("CA constraint")

  This mode is used to specify a CA certificate, or the public key of such a certificate, that MUST be found in any of the PKIX certification paths for the end entity certificate given by the server in TLS or DTLS. This certificate usage is sometimes referred to as "CA constraint" because it limits which CA can be used to issue certificates for a given service on a host. The presented certificate MUST pass PKIX certification path validation, and a CA certificate that matches the Server Public Key Resource content MUST be included as part of a valid certification path. Because this certificate usage allows both trust anchors and CA certificates, the certificate might or might not have the basicConstraints extension present.

- 1: Certificate usage 1 ("service certificate constraint")

  This mode is used to specify an end entity certificate, or the public key of such a certificate, that MUST be matched with the end entity certificate given by the server in TLS. This certificate usage is sometimes referred to as "service certificate constraint" because it limits which end entity certificate can be used by a given service on a host. The target certificate MUST pass PKIX certification path validation and MUST match the Server Public Key Resource content.

- 2: Certificate usage 2 ("trust anchor assertion")

  This mode is used to specify a certificate, or the public key of such a certificate, that MUST be used as the trust anchor when validating the end entity certificate given by the server in TLS. This certificate usage is sometimes referred to as "trust anchor assertion" and allows a domain name administrator to specify a new trust anchor -- for example, if the domain issues its own certificates under its own CA that is not expected to be in the end users' collection of trust anchors. The target certificate MUST pass PKIX certification path validation, with any certificate matching the Server Public Key Resource content considered to be a trust anchor for this certification path validation.

- 3: Certificate usage 3 ("domain-issued certificate")

  This mode is used to specify a certificate, or the public key of such a certificate, that MUST match the end entity certificate given by the server in TLS. This certificate usage is sometimes referred to as "domain-issued certificate" because it allows for a domain name administrator to issue certificates for a domain without involving a third-party CA. The target certificate MUST match the Server Public Key Resource content. The difference between certificate usage 1 and certificate usage 3 is that certificate usage 1 requires that the certificate pass PKIX validation, but PKIX validation is not tested for certificate usage 3.

5.2.9. Error Handling

The TLS / DTLS specifications offer ways to report error situations via the Alert protocol. These errors may have multiple reasons and the process to recover from them is essential for a stable LwM2M implementation. Nevertheless recovering from these errors may be challenging since many IoT devices are unattended and a failure in the establishment of a security association may result in a non-functioning device. Since a LwM2M Bootstrap-Server supervises the LwM2M Client it can support with the recovery of certain types of errors, i.e., the LwM2M Client is asked to fall back to the bootstap mode in order to obtain new credentials.

This section provides default recommended error handling. LwM2M Server Object configuration provides additional capabilities via optional resources, such as the Bootstrap on Registration Failure Resource, to define the behavior in error conditions. There is no such fallback mechanism defined for the interaction between the LwM2M Client and the LwM2M Bootstrap-Server.

Value	Alert Description	RFC	Recommendation
0	close_notify	RFC 5246	Retry
10	unexpected_message	RFC 5246	Retry
20	bad_record_mac	RFC 5246	Retry
21	decryption_failed	RFC 5246	Retry
22	record_overflow	RFC 5246	Retry
30	decompression_failure	RFC 5246	Ignore*
40	handshake_failure	RFC 5246	Retry
41	no_certificate_RESERVED	RFC 5246	Ignore*
42	bad_certificate	RFC 5246	Fail
43	unsupported_certificate	RFC 5246	Fail
44	certificate_revoked	RFC 5246	Fail
45	certificate_expired	RFC 5246	Fail
46	certificate_unknown	RFC 5246	Fail
47	illegal_parameter	RFC 5246	Retry
48	unknown_ca	RFC 5246	Fail
49	access_denied	RFC 5246	Fail
50	decode_error	RFC 5246	Retry
51	decrypt_error	RFC 5246	Retry

60	export_restriction_RESERVED	RFC 5246	Ignore*
70	protocol_version	RFC 5246	Retry
71	insufficient_security	RFC 5246	Retry
86	internal_error	RFC 7507	Retry
90	user_canceled	RFC 5246	Retry
100	no_renegotiation	RFC 5246	Retry**
110	unsupported_extension	RFC 5246	Ignore.
111	certificate_unobtainable	RFC 6066	Retry***
112	unrecognized_name	RFC 6066	Fail***
113	bad_certificate_status_response	RFC 6066	Fail***
114	bad_certificate_hash_value	RFC 6066	Fail***
115	unknown_psk_identity	RFC 4279	Fail

Table: 5.2.9.-1 TLS/DTLS Error Handling Recommendations

Legend:

- (*): This error should never occur in a proper TLS / DTLS implementation.
- (**): This error should not occur since an implementation compliant to RFC does not use the renegoation feature.
- (***): This error is only applicable when used with the appropriate extension.

The 'fail' recommendation in the table above indicates that the LwM2M Client MUST consider the handshake exchange failed and that it cannot by itself recover from the error situation. The 'retry' recommendation in the table above indicates that the LwM2M Client MUST consider the handshake failed but that the error situation MAY be transient and retrying the handshake protocol run may result in a successful completion. In both cases the LwM2M Client MUST follow the error handling procedures defined by resources in the LwM2M Server Object, if present.

An implementation may set a limit to the number of attempts to re-establish a TLS/DTLS connection and may decide to fall back to bootstrapping when a certain threshold is exceeded. Implementations should also use an exponential back-off between retransmissions. The number of retries and the retransmission timers MAY be set by resources in the LwM2M Server resources.

Note: Per RFC 6347 DTLS implementations SHOULD silently discard records with bad MACs or that are otherwise invalid.

5.3. Lower- or Higher-Layer Security with the "NoSec" mode

The LwM2M protocol MUST NOT be deployed without appropriate security. Because LwM2M may be deployed with security outside the scope of this specification (e.g. at lower layers), this may mean that neither transport-layer security (TLS/DTLS), nor application-layer security (OSCORE) is applied. In case communication security is not used at the LwM2M layer ("NoSec" mode), the deployment MUST use alternative security mechanisms.

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).

5.4. SMS Channel Security

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

Channel security for [CoAP] has been defined for the UDP transport and is based on DTLS [RFC6347].

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

LwM2M Clients supporting UDP and SMS bindings, when the SMS binding is only used for triggering, MUST support the adequate mechanism for securing CoAP over UDP. 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.
Note: Using SMS NoSec for SMS triggering could introduce "Denial of Service" (DoS).

The "Secret Key" and "Public Key or Identity" Resources are used to configure the keying material that a Client uses with a particular LwM2M Server. The SMS key parameters are stored in the order KIc, KID, SPI, TAR (KIc is byte 0). Ordering of bits within bytes SHALL follow ETSI TS 102 221, Section 3.4 "Coding Conventions" (b8 MSB, b1 LSB).

5.4.1. SMS "NoSec" Mode

It is RECOMMENDED to always use LwM2M with one of the security mechanisms described in this specification. 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 focusing first on other functions before performing end-to-end tests including security.

The use of SMS "NoSec" Mode while the Resource "OSCORE Security Mode" is present indicates that the SMS Channel is protected with OSCORE.

This security profile is also useful to support SMS triggering when all other exchanges utilize CoAP over TLS or CoAP over DTLS.

5.4.2. SMS Secured Mode

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

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

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

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

5.4.2.1. Device End-Point

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

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 text in Section 5.2. TLS/DTLS-based Security concerning the DTLS binding to CoAP is also applicable to this section.

Appendix A of [RFC7925] 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.

5.4.2.1.1. Header Definitions (for one SMS)

a) 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 byte	Effective Payload (107 byte

Figure: 5.4.2.1.1.-1 SMS header definition (no token)

Model calculation using these header definitions,

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

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

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

Figure: 5.4.2.1.1.-2 SMS header definition

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

5.4.2.1.2. Smartcard End-Point

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

The following settings MUST be applied:

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

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

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 5.4.2.1.3. SMS Secured Packet Binding for CoAP messages.

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

5.4.2.1.3. SMS Secured Packet Binding for CoAP messages

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

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

5.5. OSCORE-based Security

5.5.1. OSCORE Overview

Object Security for Constrained RESTful Environments [OSCORE] is an application layer security protocol protecting CoAP message exchanges. OSCORE is applicable to protocol messages which can be mapped to CoAP or a subset of CoAP, including HTTP and LwM2M.

The use of OSCORE with LwM2M is optional. It MAY be used for protecting the communication between a LwM2M Client and the LwM2M Bootstrap-Server. It MAY be used between a LwM2M Client and a LwM2M Server.

OSCORE MAY also be used between LwM2M endpoint and non-LwM2M endpoint, e.g. between an Application Server and a LwM2M Client via a LwM2M server. Both the LwM2M endpoint and non-LwM2M endpoint MUST implement OSCORE and be provisioned with an OSCORE Security Context as defined in [OSCORE]. The bootstrapping of the non-LwM2M endpoint is out of scope for this specification. The intermediate LwM2M node needs to map the messages from non-LwM2M nodes to LwM2M operations and responses. That mapping is also out of scope for this specification.

The use of OSCORE in CoAP is indicated with the OSCORE Option present in the CoAP message [OSCORE]. It operates by transforming (encrypting etc.) an unprotected CoAP message into a protected CoAP message using the compact secure message format COSE [RFC8152]. The protected CoAP message is sent instead of the unprotected message, and is verified and transformed back to the original message at the receiving end.

OSCORE protects the Request/Response sublayer of CoAP, but leaves the binding to the transport layer protocol unprotected. As a consequence, OSCORE does not depend on the underlying layer and can therefore be applied to CoAP over UDP, TCP, SMS as well as Non-IP, directly over IEEE 802.15.4 or 3GPP CIoT (see Appendix D). OSCORE is invariant under proxy operations translating between different transport layer protocols, performing application layer forwarding (e.g. CoAP proxies), address translations, etc. and thus enables end-to-end security through these kinds of intermediate nodes.

Not only is OSCORE applicable to CoAP-mappable messages, but the protected messages can also be transported directly in CoAP-mappable protocols. For example, with the HTTP header field "OSCORE" [OSCORE], OSCORE protected messages can be transported in HTTP, providing end-to-end security for CoAP-mappable HTTP. OSCORE also supports proxies translating between HTTP and CoAP as defined in [RFC8075], allowing a mix of HTTP and CoAP along the end-to-end path, which is useful, e.g. for firewall traversals on the Internet.

OSCORE is based on a shared symmetric key and provides security context derivation, encryption, integrity and replay protection, and a secure binding of application layer responses to requests. OSCORE enables mutual authentication between LwM2M Client and LwM2M Bootstrap-Server, and between LwM2M Client and LwM2M Server.

OSCORE is independent of security protocols at other layers, such as DTLS/TLS. If OSCORE is applied with a "Security Mode" Resource value 0-2 (corresponding to DTLS with mode "Pre-Shared Key"/"Raw Public Key"/"Certificate") the message will be protected by both OSCORE and DTLS. When OSCORE is applied with "Security Mode" Resource value 3 (corresponding to DTLS with mode "NoSec"), the message is protected by OSCORE but not DTLS.

OSCORE can be used to secure multicast requests and responses, for example providing low latency in building automation scenarios (lighting, emergency etc.) and for efficient bandwidth usage during firmware updates.

5.5.2. Cryptographic Algorithms

OSCORE is defined for use with an Authenticated Encryption with Additional Data (AEAD) algorithm and a HMAC-based Key Derivation Function (HKDF) [RFC5869]. OSCORE mandates the implementation of the AEAD algorithm AES-CCM-16-64-128 (also known as CCM*) as defined in [RFC8152], and specifies the use of HMAC with SHA-256 as default. LwM2M Clients and LwM2M Servers MAY support additional algorithms that conform to the state-of-the-art security requirements.

5.5.3. Bootstrapping

The OSCORE related Resources in the LwM2M OSCORE Object are used analogously to DTLS/TLS when applied with pre-shared keys (see 5.2.4. Bootstrapping). The OSCORE related Resources in the LwM2M OSCORE Object are used for:

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

In case of 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 the pre-shared key needs no pre-existing trust relationship between LwM2M Server(s) and LwM2M Client(s). The pre-established trust relationship exists between the LwM2M Server(s) and the SmartCard(s).

If OSCORE is used, then LwM2M Clients MUST either be provisioned for use with a LwM2M Server (manufacturer pre-configuration bootstrap mode) or else be provisioned for use with a LwM2M Bootstrap-Server. Any LwM2M Client, which supports client or server initiated bootstrap mode, MUST support bootstrapping with a strong (high-entropy) pre-shared key. The ciphersuites MUST NOT be used with a low-entropy pre-shared key or a password as pre-shared key. The LwM2M Client MUST be provisioned with a pre-shared key that is unique to a device and to the security protocol. For full interoperability a LwM2M Bootstrap-Server MUST support bootstrapping with pre-shared key.

The OSCORE security context is derived from a small set of input parameters (see Section 3.2 of [OSCORE]). With use of the OSCORE input parameters, the security contexts for sending and receiving OSCORE messages are derived as defined in [OSCORE]. The conditions for reuse of input parameters (see Section 3.3 of [OSCORE]) MUST be complied with. An implementation SHALL use a Master Salt as described in Appendix B2 of [OSCORE].

New OSCORE input parameters MAY be provisioned for use between endpoints having an existing OSCORE security context requiring the endpoints to derive new contexts. The Echo option MUST be used by the receiving endpoint when the OSCORE security context is used for the first time (e.g. Bootstrap-Request or Register) and SHOULD be used when the requested operation (e.g. Device Management Write) requires freshness that cannot be guaranteed by other means.

5.5.4. Unbootstrapping

The unbootstrapping procedure described in 5.2.5. Unbootstrapping applies also to OSCORE.

5.5.5. Endpoint Client Name

The same verification of Endpoint Client Name in the Bootstrap-Request and in the Register messages as described in 5.2.6. Endpoint Client Name applies also to OSCORE. However, when using OSCORE, the Endpoint Client Name MAY be authenticated at the application layer, by setting the "OSCORE Sender ID" Resource value (see 5.5.7.1. OSCORE Related Resources) to the Endpoint Client Name.

If the OSCORE Sender ID is not set to Endpoint Client Name, then the LwM2M Server MUST compare the received

Endpoint Client Name identifier with the OSCORE Sender ID of the LwM2M Client. 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.

5.5.6. OSCORE Roles

The OSCORE roles (Client and Server) coincide with the CoAP roles.

OSCORE specifies that each endpoint has a Sender ID and a Recipient ID. The Sender ID is used when sending a message and recepient ID when receiving a message, independently of whether the endpoint has the role of an OSCORE Client or an OSCORE Server. Hence the Sender ID of the OSCORE Client has the same value as the Recipient ID of the OSCORE Server, and the Recipient ID of the OSCORE Client has the same value as the Sender ID of the OSCORE Server.

5.5.7. Credential Types

OSCORE requires that a shared secret key, called "OSCORE Master Secret", is established in the communicating endpoints. In LwM2M version 1.1 it is assumed that the Master Secret is pre-shared.

When the Master Secret used with a LwM2M Server is provisioned to the LwM2M Client, the Master Secret is also known to the provisioning party, e.g. the LwM2M Bootstrapping Server. Applications need to ensure that this assumption is compliant with their trust model.

5.5.7.1. OSCORE Related Resources

The use of OSCORE between LwM2M Client and LwM2M Server, or between LwM2M Client and LwM2M Bootstrap-Server is indicated with the optional "OSCORE Security Mode" Resource in the Security Object description.

- If present, the "OSCORE Security Mode" Resource MUST contain the link to an Instance of the LwM2M OSCORE Object.

The presence of the "OSCORE Security Mode" Resource in a Security Object Instance requires the following resources of the LwM2M OSCORE Object to be populated:

- The "OSCORE Master Secret" Resource MUST be used to store the Master Secret, as defined in [OSCORE]. The OSCORE Client and OSCORE Server use the same Master Secret. Recommendations for generating random keys are provided in [RFC4086].
- The "OSCORE Sender ID" Resource MUST be used to store the Sender ID of the LwM2M Client as an UTF-8 string, as defined in [OSCORE].
- The "OSCORE Recipient ID" Resource MUST be used to store the Recipient ID of the LwM2M Client as an UTF-8 string, as defined in [OSCORE].

Additionally, the following resources of the LwM2M OSCORE Object MAY be populated:

- The "OSCORE AEAD Algorithm" Resource MUST be used to store the AEAD Algorithm, as defined in [OSCORE].
- The "OSCORE HMAC Algorithm" Resource MUST be used to store the HMAC Algorithm used with HKDF, as defined in [OSCORE].

- The "OSCORE Master Salt" Resource MUST be used to store the Master Salt, as defined in [OSCORE].

5.6. Identification of Blockwise Requests

The specification of Blockwise [CoAP_Blockwise] is vulnerable to interchange of blocks between different requests to the same resource [CoAP_ERT]. The attack may be performed when the replay window size of the the security protocol is greater than 1 even if the requests are not inverleaved and the attack applies both to DTLS and OSCORE. The attack does not apply when a connection-oriented transport, like CoAP over TCP is used, or when a replay window size of 1 is selected with DTLS.

A solution is to use the CoAP Request-Tag Option [CoAP_ERT] for unique tagging of requests of a certain scope. The Request-Tag is analogous to the CoAP E-Tag Option, but tags requests instead of responses.

LwM2M Clients and LwM2M Servers supporting Blockwise SHOULD implement the CoAP Request-Tag Option.

5.7. Freshness

A LwM2M Client and LwM2M Server MUST be able to verify the freshness of certain LwM2M operations.

LwM2M operations may have freshness requirements that are not possible to guarantee by protecting individual messages such as with DTLS Record Layer or OSCORE. This is accentuated with unreliable transport. Since datagram transport does not provide reliable or in-order delivery of data, DTLS and OSCORE preserves this property. Although duplicate messages are rejected through the use of anti-replay mechanisms, unordered delivery is still allowed, e.g. using a sliding receive window. As a consequence maliciously delayed message are accepted as long as they fall within the window.

For example, a Write operation, maliciously blocked from reaching the LwM2M Client at one time, may under these circumstances be successfully injected at a later time, potentially overwriting a more recent Write operation.

Operations protected by a security protocol with keys derived from a TLS/DTLS handshake are at least as fresh as the handshake. However, frequent use of the handshake protocols may be prohibitive in constrained environments. In order to avoid unnecessary processing, a more lightweight solution to verify freshness is provided by the CoAP Echo Option [CoAP_ERT], illustrated with the example above: The LwM2M Client, receiving a Write operation of uncertain freshness may respond with an error message containing an Echo option including a random nonce as value. The LwM2M Server receiving the error response to a valid Write operation retransmits the request with the Echo option and value included. The LwM2M Client receiving a request with an Echo option verifies that the nonce corresponds to a recent request, and only in that case performs the operation (for details, see Section 2 of [CoAP_ERT]).

Applications need to understand the freshness requirements of the operations both in LwM2M Client and LwM2M server. The LwM2M implementation SHOULD enable timely freshness verifications to be performed without unnecessary overhead. For interoperability both LwM2M Client and LwM2M Server SHOULD implement the CoAP Echo option.

The attack does not apply when a connection-oriented transport, like CoAP over TCP is used, or when a replay window size of 1 is selected with DTLS.

6. CoAP Transport Binding

This section defines the CoAP transport binding used by LwM2M interfaces.

6.1. Features

The CoAP transport binding utilizes the following features:

- CoAP, as defined in [CoAP], MUST be supported by the LwM2M Client and the LwM2M Server.
- CoAP over TCP/TLS SHOULD be [CoAP-TCP] be used to enable improved firewall and NAT traversal capabilities.
- CoAP Observe, as defined in [Observe], MUST be supported by the LwM2M Client and the LwM2M Server for the Information Reporting interface. Non-Confirmable messages MAY be used by a Client for sending Information Reporting notifications as per [Observe].
- CoAP Blockwise transfer for CoAP [CoAP_Blockwise] 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 [RFC7252] 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 [CoAP_Blockwise] was specifically designed to lift this limitation in order to transfer large payloads larger than ~64 KiB via CoAP, such as firmware images. [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.
- The CoAP OSCORE Option MAY be used to enable OSCORE.
- The CoAP Request-Tag Option [CoAP_ERT] SHOULD be used to detect interchange of blocks between different blockwise requests to the same resource over unreliable transport.
- The CoAP Echo Option [CoAP_ERT] SHOULD be used to enable lightweight freshness verifications.

6.2. Firewall Traversal

For a firewall to support LwM2M using CoAP over UDP and/or CoAP over DTLS it MUST at a minimum be configured to allow outgoing packets to destination port 5683 (for CoAP over UDP), and port 5684 (for CoAP over DTLS), and allow incoming UDP/DTLS packets back to the source address/port of the outgoing UDP packet for a period of at least 240 seconds.

While Queue Mode is not necessary for firewall configuration it should be noted that LwM2M Clients may enable Queue Mode for example to lower power consumption for a battery powered device.

For those deployments where changes to firewall configurations are not possible CoAP over TCP/TLS SHOULD be used instead. Firewall rule settings for CoAP over TCP/TLS are similar to those described above for CoAP over UDP/DTLS since the port numbers are equivalent, i.e., port 5683 (for CoAP over TCP), and port 5684 (for CoAP over TLS).

6.3. NAT Traversal

A NAT exhibits a behavior similar to a stateful packet filtering firewall where incoming packets are only allowed to traverse the NAT after the client behind the NAT initiated an outgoing connection first. The NAT creates a NAT binding based on the outgoing communication attempt, which is re-used for any packets from the device outside the NAT (i.e., typically the server).

Where NATs are present along the communication path, CoAP over TCP leads to different NAT traversal behavior than CoAP over UDP. NATs often calculate expiration timers based on the transport layer protocol being used by application protocols. Many NATs maintain TCP-based NAT bindings for longer periods based on the assumption that a transport layer protocol, such as TCP, offers additional information about the session lifecycle. UDP, on the other hand, does not provide such information to a NAT and timeouts tend to be much shorter, as described in [CoAP-TCP].

Queue Mode may help with NAT traversal by allowing a LwM2M Server to send messages to the LwM2M Client only after the LwM2M Client has initiated the communication to the LwM2M server first. This ensures that NAT bindings exist that allow incoming packets to the LwM2M Client.

6.4. 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 using CoAP layer MUST be Confirmable CoAP messages, except as follows:

- "Notify" which may be a Non-Confirmable CoAP message
- Triggering an execution on a resource can be a Non-Confirmable CoAP message

6.4.1. 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 RFC 6690 [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.

6.4.2. Bootstrap Interface

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

The Client Bootstrap operation is initiated by the LwM2M Client itself. In addition, this operation can be requested by an authorized LwM2M Server executing the "Bootstrap-Request Trigger" Resource of a Server Object Instance, or even by a proprietary mechanism (e.g. based on SMS). Note: the execution of a Bootstrap-Request Trigger Resource by a LwM2M Server in a LwM2M Client is performed through an already established registration and is therefore covered by the access rights mechanisms.

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

In "Client Initiated Bootstrap" mode, when the Bootstrap-Server receives Bootstrap-Request operation, the Bootstrap-Server can perform Write, Discover, Read and/or Delete operations. The Delete operation targets an Object or an Object Instance, the Discover operation targets an Object, a Write operation targets Object, Object Instance or a Resource, while the Read operation is limited to read a single Instance or all Instances of the Access Control Object (an error is returned otherwise). The Read, 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 sends finish message by sending CoAP POST to "/bs" location path with empty payload.

If OSCORE is used to secure Client Bootstrap then the Bootstrap-Server receiving the Bootstrap-Request operation MUST use the Echo Option [CoAP_ERT].

Operation	CoAP Method	URI	Success	Failure
Bootstrap-Request	POST	/bs?ep={Endpoint Client Name}&pct={Preferred Content Format}	2.04 Changed	4.00 Bad Request, 4.15 Unsupported content format

Read	GET Accept:Content Format ID: TLV, CBOR or JSON	/{Object ID} in LwM2M 1.1 Object ID MUST be '2' (Access Control Object)	2.05 Content	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method not Allowed, 4.06 Not Acceptable
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: 6.4.2.-1 Operation to Method and URI Mapping (Bootstrap Interface)

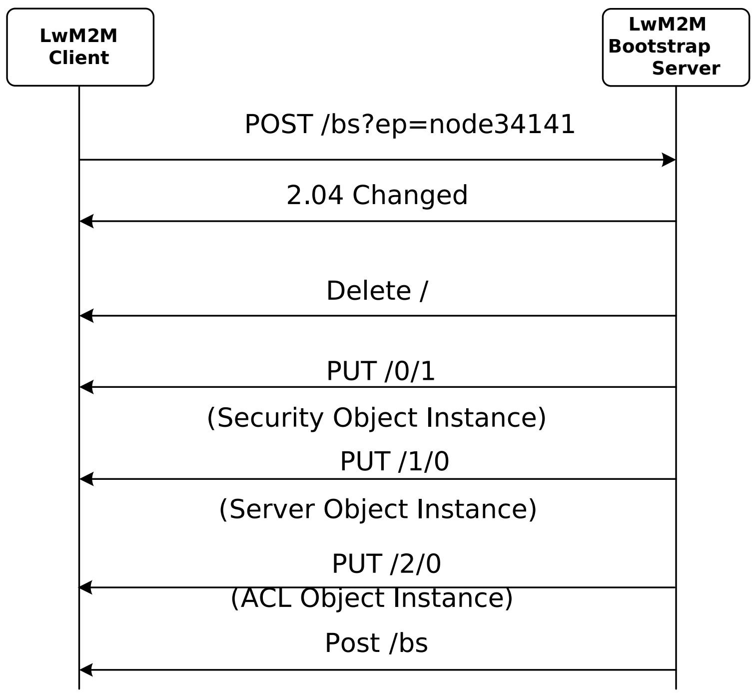

Figure: 6.4.2.-1 Example of Client initiated Bootstrap exchange

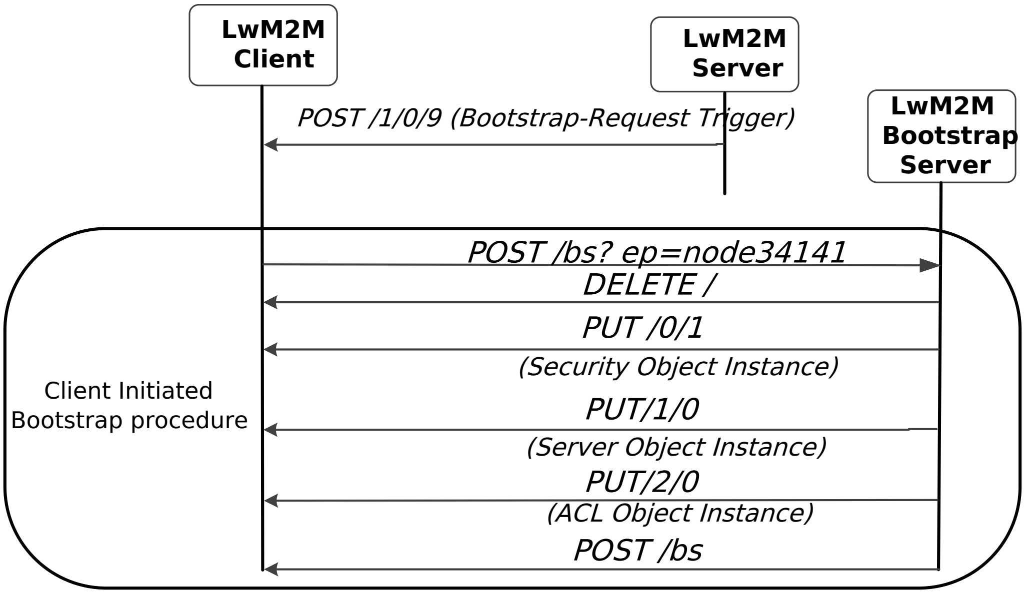

Figure: 6.4.2.-2 Example of Server initiated Bootstrap exchange

6.4.3. Registration Interface

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

Registration is performed by sending a CoAP POST to the LwM2M Server URI /rd, with registration parameters passed as query string parameters as per Table: 6.4.3.-1 Operation to Method and URI Mapping (Registration Interface) and Object and Object Instances included in the payload as specified in [LwM2M-CORE]. 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 TLS/DTLS handshake is started, or in NoSec mode when the LwM2M Client IP address changes, the Client MUST register again to the LwM2M Server.

If OSCORE is used to secure the Registration then the Echo Option [CoAP_ERT] MUST at least be used with the first operation.

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}&lwm2m={version}&b={binding}&Q&sms={MSISDN}	2.01 Created	4.00 Bad Request, 4.03 Forbidden, 4.12 Precondition Failed

Update	POST	/{location}?lt={Lifetime}&b={binding}&Q&sms={MSISDN}	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: 6.4.3.-1 Operation to Method and URI Mapping (Registration Interface)

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.

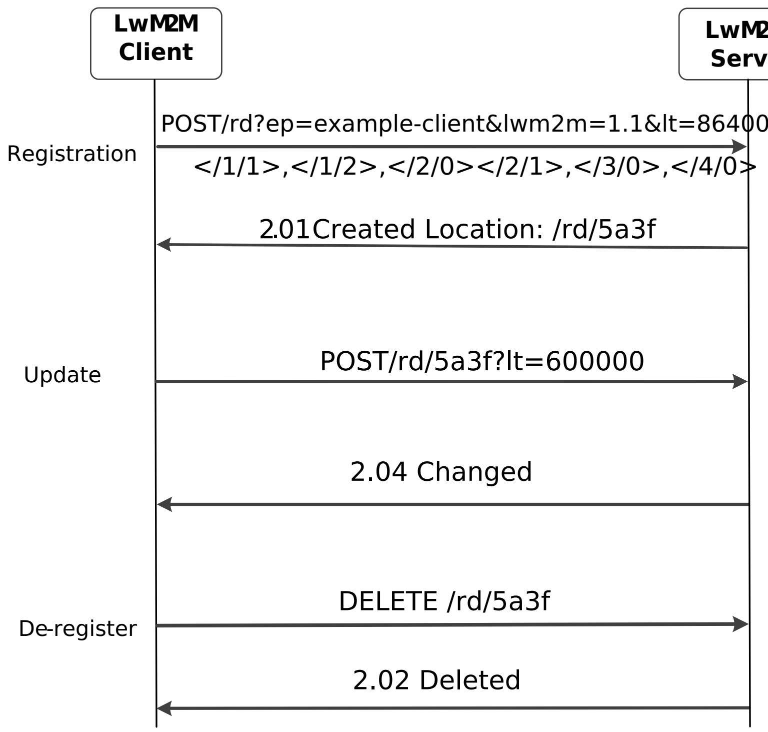

Figure: 6.4.3.-1 Example register, update and de-register operation exchanges (shorthand in \[CoAP\] example style, actual messages using CoAP binary headers

6.4.4. Device Management & Service Enablement Interface

The Device Management & Service Enablement Interface is used to access a Resource, a Resource Instance, 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 Instance of a Multiple-Instance Resource is identified by the path /{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID}. When the rules expressed in [LwM2M-CORE] and related to the path in the command on this LwM2M Interface, are not respected, the error code 4.05 MUST be returned as response code.

An Object Instance, a Resource or a Resource Instance are read by sending a CoAP GET to the corresponding path. The response includes the value in the corresponding format according to the specified Content-Format (see [LwM2M-CORE]).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 (error code 4.06).

Note that the response payload may be empty, for instance when performing a Read operation on an Object with no Object Instance. In this case the response code is still 2.05 Content.

An Object Instance, a Resource or a Resource Instance are written by sending a CoAP PUT request to the corresponding path. The request includes the value to be written in the corresponding Plain Text, Opaque, TLV, CBOR or JSON format according to the Content-Format option which MUST be specified [CoAP]. In addition, an Object Instance can be written by sending a CoAP POST request; in that case the specified Content-Format MUST be one of the TLV, CBOR or JSON formats. A CoAP PUT is used for the Replace and CoAP POST is used for Partial Update mechanism of the "Write" operation as described in [LwM2M-CORE].

The Write request MUST be rejected :

- when the specified Content-Format is not supported by the LwM2M Client (error code 4.15)
- when a requested operation on a Multiple-Instance Resource is not authorized by the LwM2M Client (error code 4.01) as described in [LwM2M-CORE].

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 as described in Appendix D of [LwM2M-CORE].

The list of arguments can be empty, have one argument, or have multiple arguments. Multiple arguments are separated by a comma. The syntax of the arguments is provided in [LwM2M-CORE].

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, CBOR 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 [LwM2M-CORE]. 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 or assigned attributes, from the LwM2M Client (see [LwM2M-CORE] for more details on DISCOVER command). With the "Write-Attributes" operation one or more Attributes can be written at a time. When several Attributes are specified in the same "Write-Attributes" command, they MUST be consistent according to the rules defined in [LwM2M-CORE], otherwise the "Write-Attributes" command MUST be rejected (4.00 Bad Request).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 separate set of configured Attributes for each individual LwM2M Server.

A "Write-Attributes" command specifies which value is set to which Attribute and at which level (Object / Object Instance / Resource / Resource Instance). 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 (see [LwM2M-CORE]).

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 [LwM2M-CORE]).
3. Write-Attributes /3/0?pmin=10 means that all Resources of Object 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&st=10 means the Battery Level will be notified to the Server when:
    1. old value is 45 and new value is 50 due to gt condition
    2. old value is 38 and new value is 49 due to both gt and step conditions

Note: if old value is 48 and new value is 42, it is not considered as satisfying any condition.

5. Write-Attributes /3/0/9?lt=20&gt=85&st=10 means the Battery Level will be notified to the Server when:
    1. old value is 75 and new value is 90 due to both gt and step conditions
    2. old value is 50 and new value is 10 due to both lt and step conditions
    3. old value is 87 and new value is 99 due to step condition

Note: if old value is 17 and new value is 24, this is not considered for any action by the client as no condition is satisfied

The value for an Object, an Object Instance, a Resource or a Resource Instance can also be sent by a LwM2M Client in an unsolicited manner to the LwM2M Server by using the "Send" command. That command is carried out by sending a CoAP POST request to a predetermined path on the LwM2M Server. That path is /dp, standing for "data push". The "Content-Format" CoAP option (header) MUST be set to inform the LwM2M Server of the format of the payload sent in that CoAP POST request. The content format MUST be either JSON or CBOR format described in [LwM2M-CORE].

As example:

Reporting the location of the LwM2M Client and the radio signal strength:

```
CoAP POST Uri-Path:"dp" Content-Format:application/senml+json
Payload:
[
  {"n":"/6/0/0", "v":43.61092},
  {"n":"/6/0/1", "v":3.87723},
  {"n":"/4/0/2", "v":-49}
]
```

Operation	CoAP Method and Content Formats	Path	Success	Failure
Read	GET Accept: see [LwM2M-CORE]	/{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID}	2.05 Content	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable
Read-Composite	FETCH Content Format: CBOR or JSON Accept: CBOR or JSON (see [LwM2M-CORE])	URI paths are provided in request payload	2.05 Content	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable, 4.15 Unsupported Content-Format
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 or POST Content Format: see [LwM2M-CORE]	/{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID} /{Object ID}/{Object Instance ID}	2.04 Changed, 2.31* Continue	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed, 4.06 Not Acceptable, 4.08* Request Entity Incomplete, 4.13* Request entity too large
Write-Attributes	PUT	/{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID}?pmin={minimum period}&pmax={maximum period}&gt={greater than}&lt={less than}&st={step}&epmin={minimum evaluation period}&epmax={maximum evaluation period}	2.04 Changed	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Write-Composite	iPATCH Content Format: CBOR or JSON (see [LwM2M-CORE])	URI path and a new value for each of resource to be written to is provided in request payload	2.04 Changed	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable
Execute	POST Content Format: text/plain or none	/{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: CBOR, JSON, or TLV (see [LwM2M-CORE])	/{Object ID}	2.01 Created	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable
Delete	DELETE	/{Object ID}/{Object Instance ID}	2.02 Deleted	4.00 Bad Request, 4.01 Unauthorized, 4.04 Not Found, 4.05 Method Not Allowed
Send	POST Content Format: CBOR or JSON (See [LwM2M-CORE])	/dp	2.04 Changed	4.00 Bad Request, 4.04 Not Found

Table: 6.4.4.-1 Operation to Method and URI Mapping (Device Management & Service Enablement Interface)

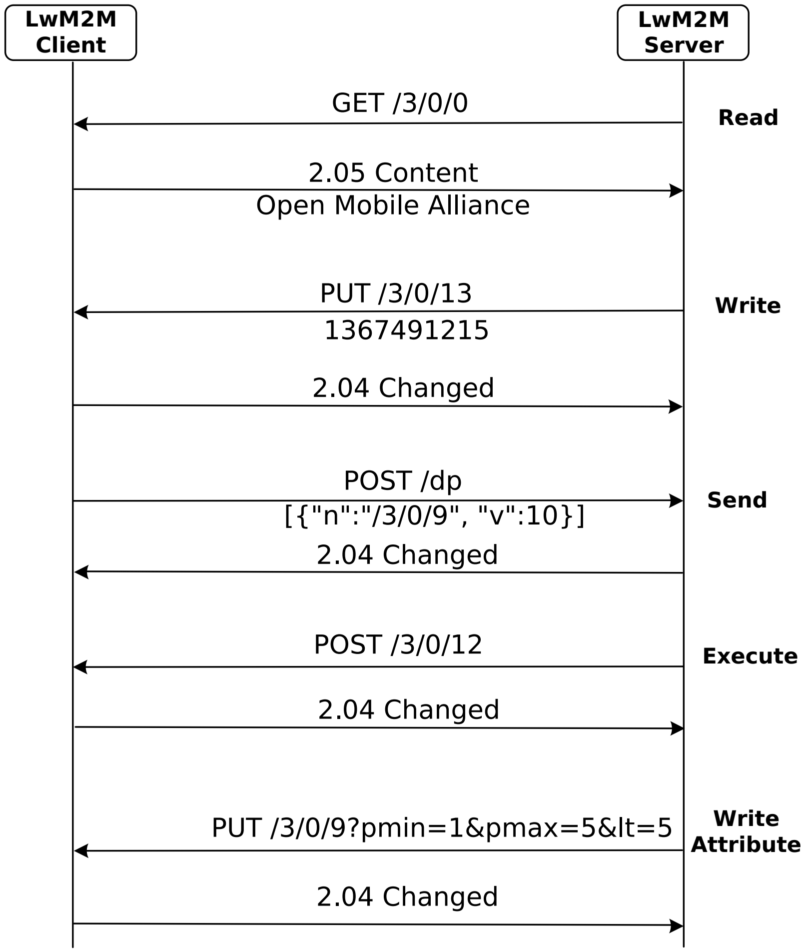

Figure: 6.4.4.-1 Example of Device Management & Service Enablement interface exchanges

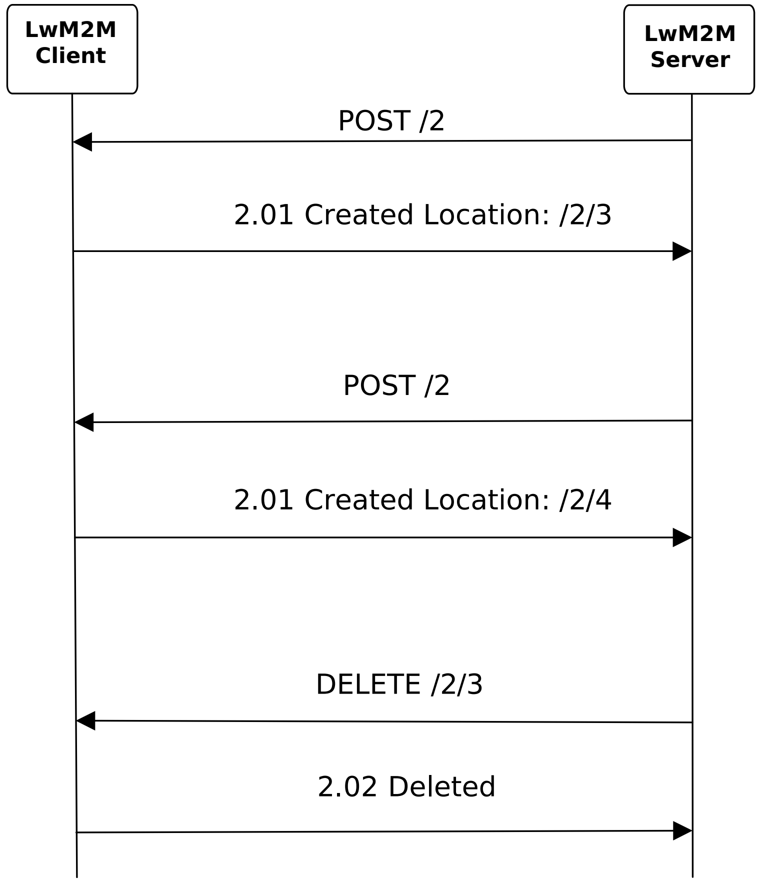

Figure: 6.4.4.-2 Example of Object Creation and Deletion

6.4.5. Information Reporting Interface

Periodic and event-triggered reporting about Resource values from the LwM2M Client to the LwM2M Server is achieved through CoAP Observation [OBSERVE]. This simple mechanism allows the LwM2M Server to send a GET request with Observe option =0 for an Object, Object Instance, Resource or Resource Instance 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, Resource Instance 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 of its notifications using the "Write-Attributes" operation (see [LwM2M-CORE]).

Operation	CoAP Method	Path	Success	Failure

Observe	GET with Observe option = 0	/{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID}	2.05 Content with Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Observe-Composite	FETCH with Observe option = 0, Content Format ID: CBOR or JSON, Accept: Content Format ID: CBOR or JSON (see [LwM2M-CORE])	URI paths for resources to be observed are provided in request payload	2.05 Content with Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed, 4.15 Unsupported Content-Format
Cancel Observation	GET with Observe option= 1	/{Object ID}/{Object Instance ID}/{Resource ID}/{Resource Instance ID}	2.05 Content without Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed
Cancel Observation-Composite	FETCH with Observe option= 1	URI paths for exactly the same resources listed in the "Observe-Composite" operation that is being canceled are provided in request payload	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: 6.4.5.-1 Operation to Method and URI Mapping (Information Reporting Interface)

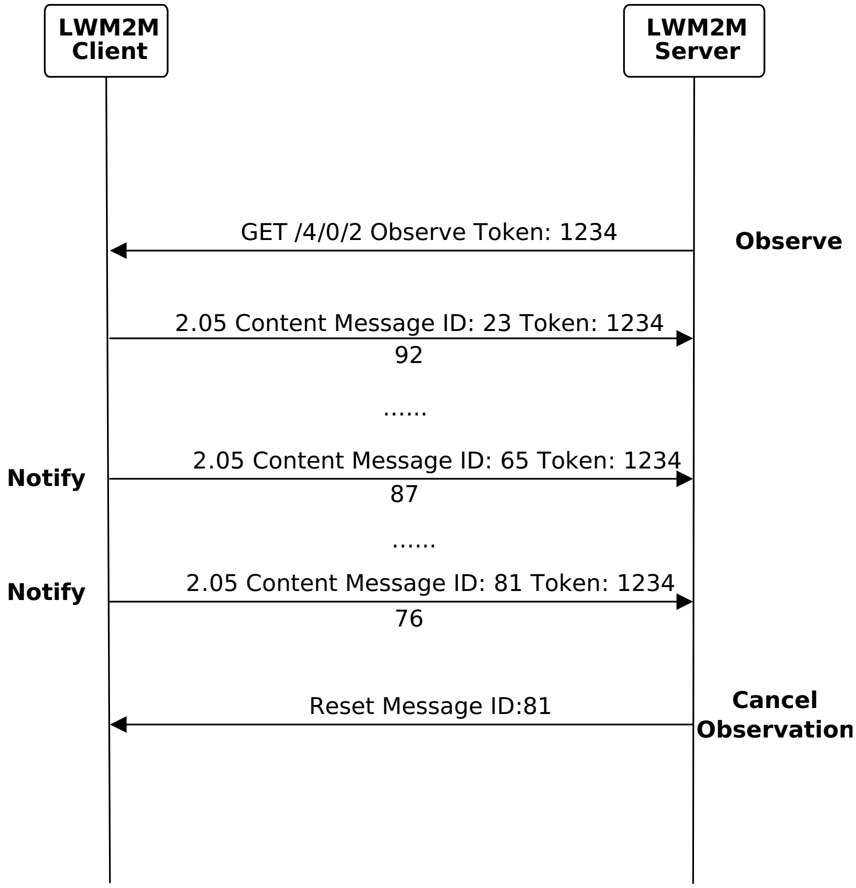

Figure: 6.4.5.-1 Example of an Information Reporting exchange

6.5. Queue Mode Operation

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

The LwM2M Client indicates support of Queue Mode in the registration to the LwM2M Server. When the LwM2M Client is not online, the LwM2M Server does not immediately send downlink requests, 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. 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 TLS/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 TLS/DTLS session resumption.
4. When the LwM2M Server receives a message from the Client, it 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.

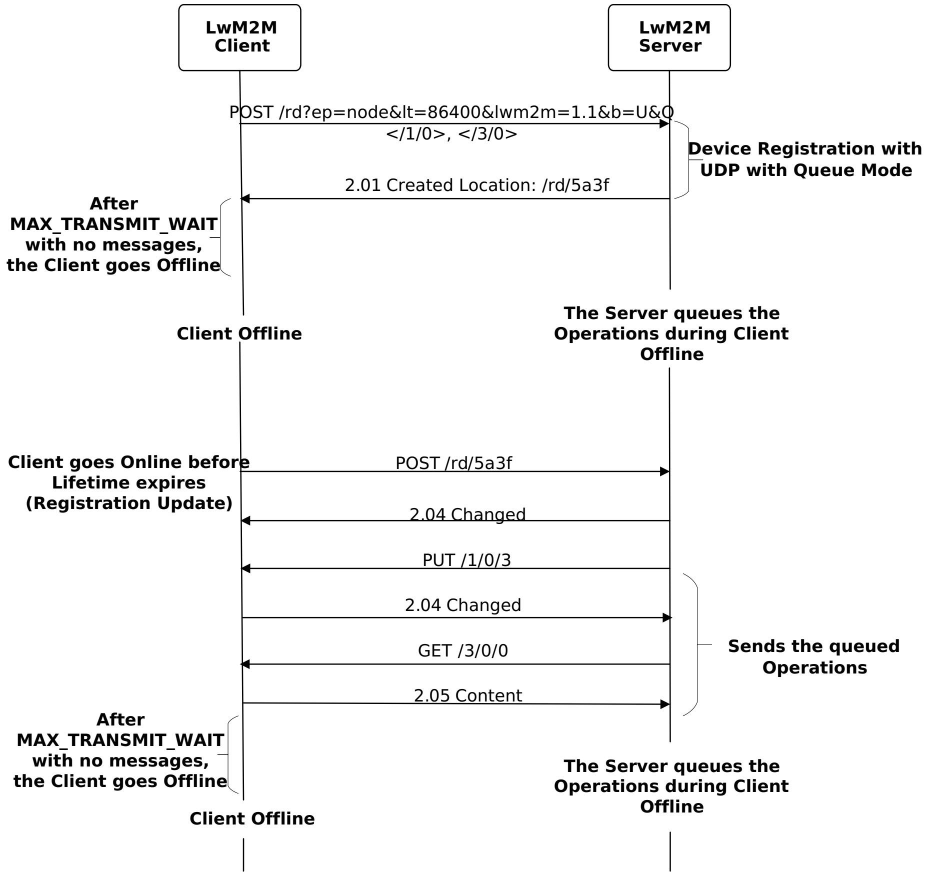

Figure: 6.5.-1 Example of Device Management & Service Enablement interface exchanges for Queue Mode

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

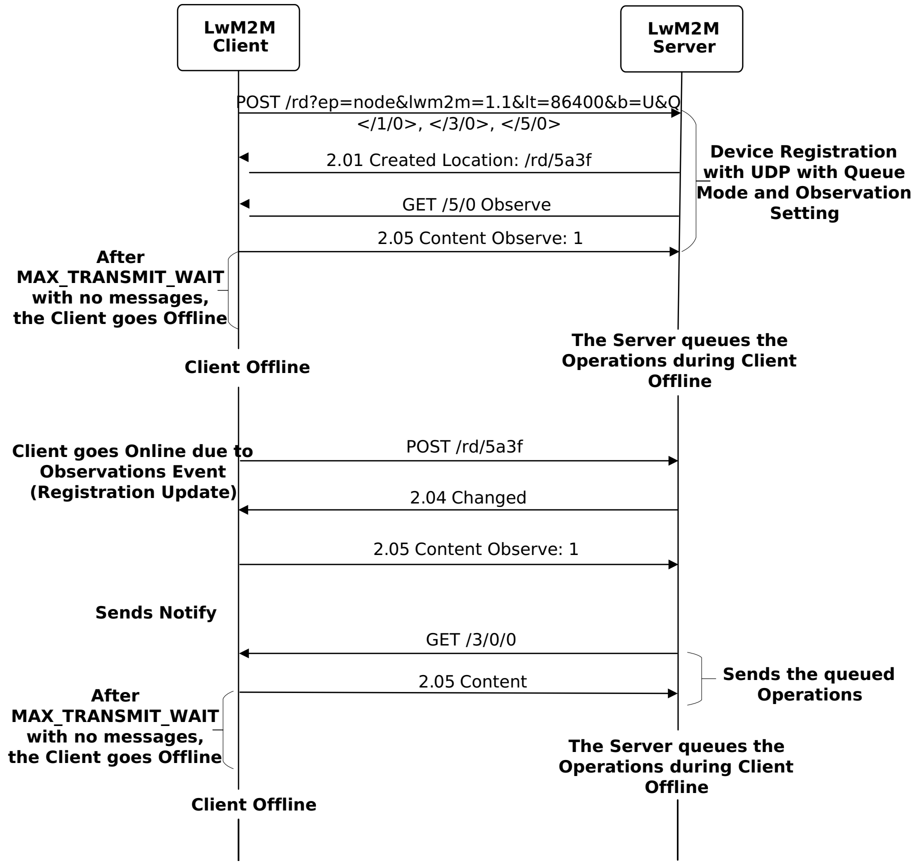

Figure: 6.5.-2 Example of an Information Reporting exchange for Queue Mode

6.6. SMS Registration Update Trigger Mechanism

When the LwM2M Client has registered to a LwM2M Server, the LwM2M Server can make the LwM2M Client update its registration by executing the Registration Update Trigger Resource in the matching Server Object Instance (see [LwM2M-CORE] Appendix).

When the LwM2M Client has registered with a Current Transport Binding different than "S", if the Trigger Resource in the matching Server Object Instance exists and is true, the LwM2M Server MAY send this trigger message over SMS. The LwM2M Client informs the LwM2M Server of its capability to receive trigger messages over SMS by including the SMS Number parameter in its registration message as defined in [LwM2M-CORE].

Below is an example flow 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 connect to the LwM2M server, where "x" represents the right instance pointing to the server. Optionally, a transport can be included in the Registration Update Trigger. Upon receiving POST /1/x/8?0=U, the LwM2M Client MAY use this transport binding to reconnect with the LwM2M Server. In this example, U refers to the UDP Binding. The possible values for binding Resource is listed in 5.3.1.1.

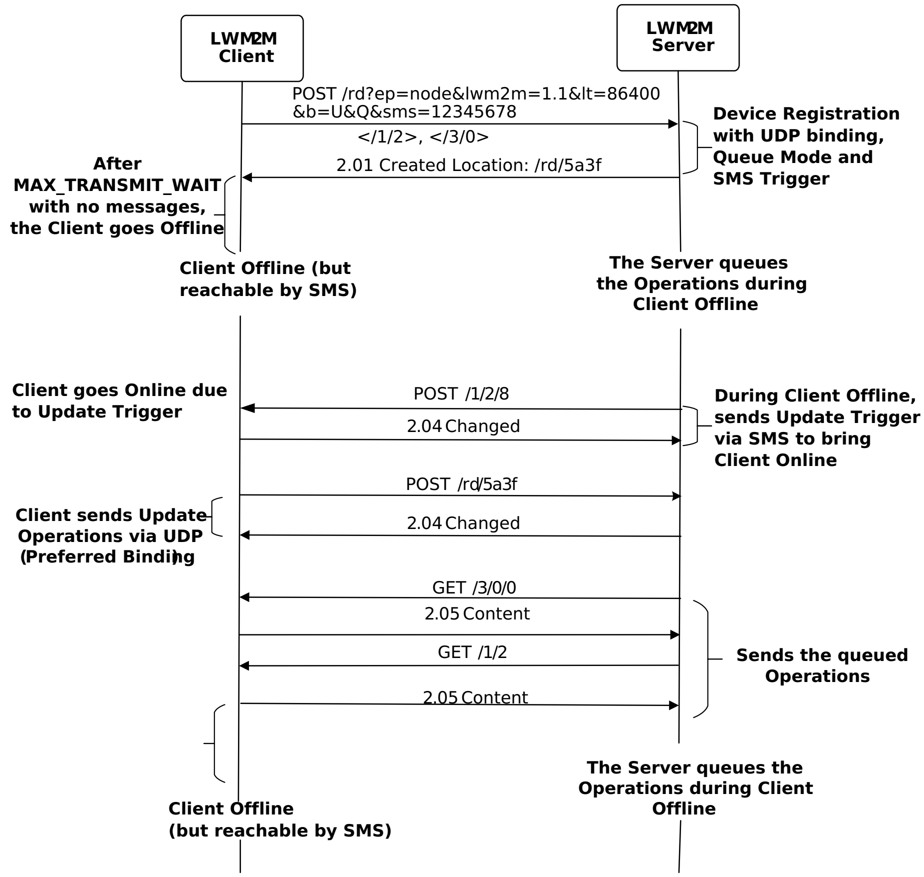

Figure: 6.6.-1 Example of Device Management & Service Enablement interface exchanges for Queue Mode with SMS Registration Update Trigger

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-Request	2.04 Changed	Bootstrap-Request is completed successfully
	4.00 Bad Request	Unknown Endpoint Client Name
Read	2.05 Content	"Read" operation is completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.01 Unauthorized	Access Right Permission Denied
	4.04 Not Found	URI of "Read" operation is not found
	4.05 Method Not Allowed	Target is not allowed for "Read" operation
	4.06 Not Acceptable	None of the preferred Content-Formats can be returned
Write	2.04 Changed	"Write" operation is completed successfully
	4.00 Bad Request	The format of data to be written is different
	4.15 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

Table: 6.6.-1 Response Codes: Bootstrap Interface

Operations	Available CoAP Response Codes	Reason Phrase
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.
	4.09 Conflict	The Client registration conflicts with the LwM2M Server's configuration.
	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

Table: 6.6.-2 Response Codes: Client Registration Interface

Operations	Available CoAP Response Codes	Reason Phrase
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
	2.02 Deleted	"Delete" operation is completed successfully

Delete	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
Send	2.04 Changed	"Send" operation completed successfully
	4.00 Bad Request	Undetermined error occurred
	4.04 Not Found	Reported Object was not registered to the LwM2M Server

Table: 6.6.-3 Response Codes: Device Management and Service Enablement Interface

Operations	Available CoAP Response Codes	Reason Phrase
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: 6.6.-4 Response Codes: Information Reporting Interface

Table: 6.6.-4 Response Codes: Information Reporting Interface

If any operation in Table 2 to Figure 8 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.

6.7. CoAP Transport Bindings

The LwM2M Server and the LwM2M Client MUST support the UDP binding, specified in Section 6.7.1. UDP Binding, and the LwM2M Server SHOULD support the SMS binding specified in Section 6.7.3. SMS Binding. The LwM2M Client MAY support other bindings defined in this specification.

6.7.1. UDP Binding

The CoAP binding for UDP is defined in [CoAP]. The protocol has an IANA registered scheme of coap:// and a default port of 5683. The UDP binding is used in NoSec (no security) mode. The CoAP binding for DTLS is defined in [CoAP]. The URI scheme is coaps:// and the default port is 5684.

Reliability of CoAP messages over the DTLS transport is provided by the built-in retransmission mechanism of CoAP.

6.7.2. TCP Binding

The CoAP binding for TCP is defined in [CoAP_TCP]. The protocol has an IANA registered scheme of coap+tcp:// and a default port of 5683. The TCP binding is used in NoSec (no security) mode. The CoAP binding for TLS is defined in [CoAP_TCP]. The URI scheme for CoAP over TLS is coaps+tcp:// and the default port is 5684.

6.7.3. 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. A 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.

6.7.4. LoRaWAN Binding

When using LwM2M over LoRaWAN, the LoRaWAN Endpoint MUST be the LwM2M Client. The LwM2M Server and the LwM2M Bootstrap-Server MUST be LoRaWAN Application Servers as defined by [LoRaWAN].

The LoRaWAN binding is used in NoSec (no security) mode. Security is provided by the LoRaWAN transport.

The Server URI MUST be in the form "lorawan://{port}", *port* being the FPort between 1 and 255 as defined in [LoRaWAN].

Due the limited bandwidth of the LoRaWAN transport, the "register" operation has different parameter requirements on this transport.

Parameter	Required	Default Value	Notes
Endpoint Client Name	No	DevEUI	This parameter is omitted if the endpoint name is identical to the endpoint DevEUI
Lifetime	No	2 592 000 (30 days)	
LwM2M Version	No	1.1	This parameter is omitted if the LwM2M enabler version is 1.1
Binding Mode	No		By nature, the LoRaWAN network is equivalent to Queue Mode.
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	No		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: 6.7.4.-1 LoRaWAN binding registration parameters

The Objects and Object Instances list MAY be omitted in the first registration attempt. If the LwM2M Server can not retrieve this list by out-of-band means (e.g. preconfiguration of the endpoint on the Application Server), it MUST reply to the registration message with a 4.09 (Conflict) code. In this case, the LwM2M Client SHOULD retry to register with a registration message that MUST include the Objects and Object Instances list.

CoAP is used over LoRaWAN in this transport binding by placing a CoAP message in the LoRaWAN packet payload. Piggybacked responses MUST be used.

Due to the high latency of the LoRaWAN network, its asymmetric nature and the intermittent availability of the LoRaWAN endpoints, CoAP retransmission is disabled for this binding and is replaced by the logic defined in Appendix CoAP over LoRaWAN (Normative).

6.7.5. CIoT Binding

CoAP can be used over 3GPP CIoT [3GPP-TR_23.720] as part of IP and non-IP data transmissions over User Plane and Control Plane. This is described in more detail in Appendix D.

Appendix A. Change History (Informative)

A.1 Approved Version 1.1 History

Reference	Date	Description
OMA-TS-LightweightM2M_Transport-V1_1-20180710-A	10 Jul 2018	Status changed to Approved by DMSE WG on 10 Jul 2018.

Table: A.1-1 Approved Version 1.1 History

Appendix B. Static Conformance Requirements (Normative)

Static Conformance Requirements are addressed in the LwM2M ETS.

Appendix C. CoAP over LoRaWAN (Normative)

This appendix defines the retransmission logic of CoAP CON messages over a LoRaWAN network.

C.1 LoRaWAN overview

A LoRaWAN network is composed of Endpoints communicating with Application Servers throught a Network Server. The role of the Network Server is to dispatch messages from the Endpoints to the revelant Application Server and to buffer messages from the Application Server to the Endpoint until the Endpoint is available for reception. A "Class A" Endpoint can receive messages only during RX windows opened after transmitting a message. "Class B"and "Class C" Endpoints are able to receive at specified periodicity. As LoRaWAN uses frequency in an unlicensed spectrum, Endpoints must respect a duty cycle which varies from region to region.

A LoRaWAN network features two kind of messages:

- confirmable messages which are retransmitted up to 8 times until an acknowledgement message is received.
- unconfirmable messages which are sent only once with no guarantee of reception on the other end.

C.2 Configuration

MAX_RETRANSMIT is set to 4.

ACK_TIMEOUT is set to 300 seconds.

C.3 Retransmission logic

C.3.1 Using LoRaWAN confirmable messages

C.3.1.1 Requests originating from the Endpoint

1. The Endpoint sends a CoAP CON message. If the matching CoAP ACK or RST is received during its RX windows, the exchange is successful.
2. Otherwise, when the Endpoint is available, it opens a RX window by sending a LoRaWAN packet. If the Endpoint has no data scheduled for transmission, it sends a CoAP Empty message instead.

The Endpoint repeats step 2. MAX_RETRANSMIT times until the matching CoAP ACK or RST is received.

The LwM2M Server MUST reply to CoAP CON messages as soon as possible.

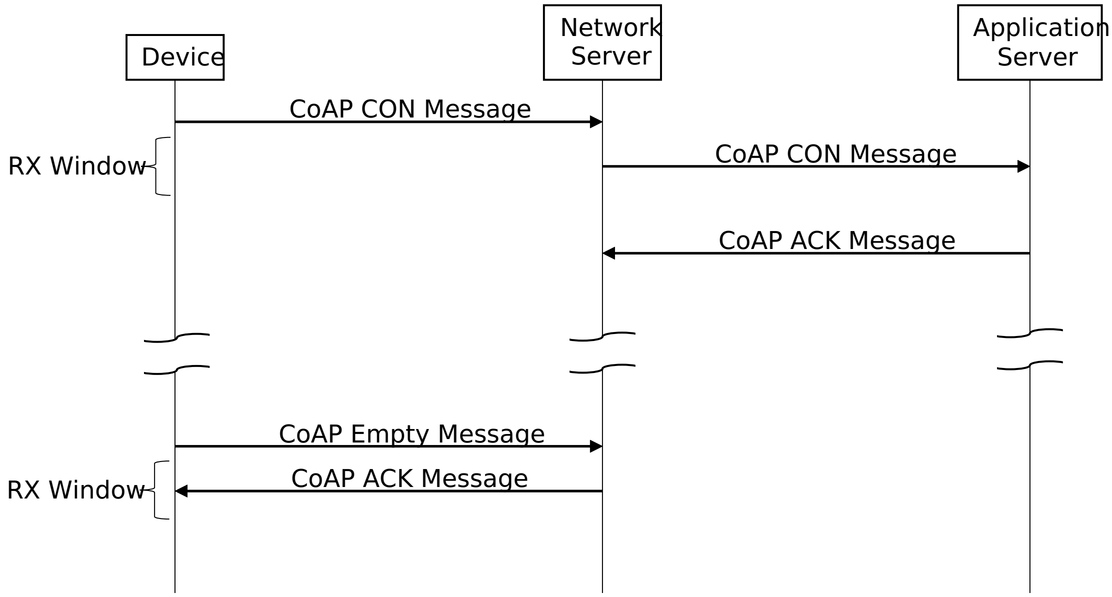

Figure: C.3.1.1-1 Example of CoAP request from Endpoint using LoRaWAN confirmable messages

C.3.1.2 Requests originating from the Application Server

1. The Application Server sends a CoAP CON message.
2. The Network Server notifies the Application Server when the CoAP message was sent to the Endpoint or when the sending failed.
3. If the sending was successful, the Application server waits for the CoAP ACK or RST message from the Endpoint.

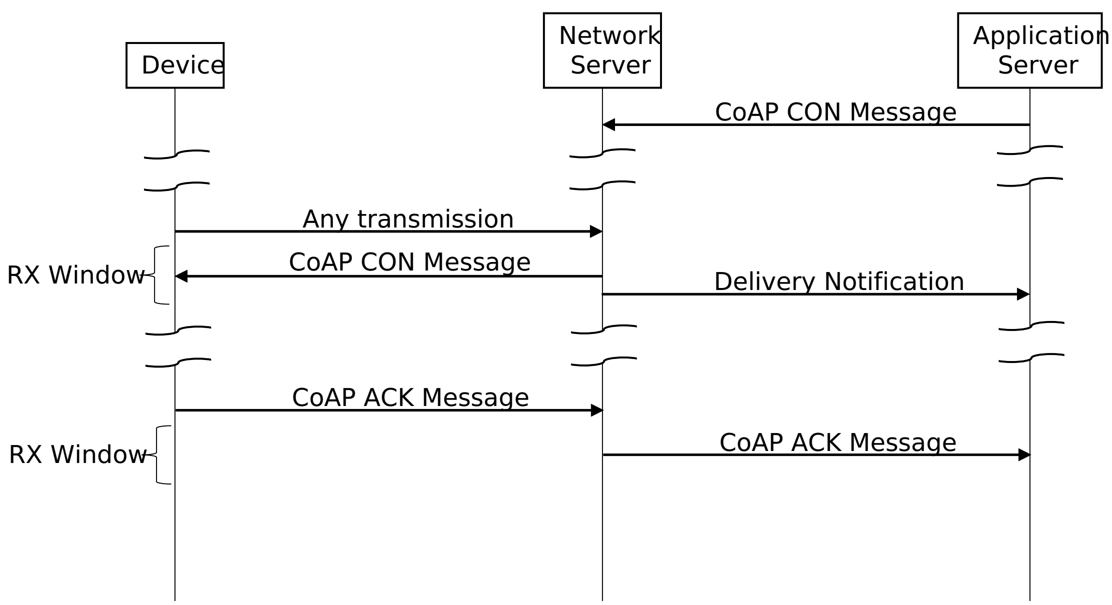

Figure: C.3.1.2-1 Example of CoAP request from Application Server using LoRaWAN confirmable messages

C.3.2 Using LoRaWAN unconfirmable messages

C.3.2.1 Requests originating from the Endpoint

1. The Endpoint sends a CoAP CON message. If the matching CoAP ACK or RST is received during its RX windows, the exchange is successful.
2. Otherwise, when the Endpoint is available, it opens a RX window by sending a LoRaWAN packet. If the Endpoint has no data scheduled for transmission, it sends a CoAP Empty message instead. If the matching CoAP ACK or RST is received during its RX windows, the exchange is successful.
3. Otherwise, when the Endpoint is available, it resends the initial CoAP CON message. If the matching CoAP ACK or RST is received during its RX windows, the exchange is successful.

The Endpoint repeats steps 2. and 3. **MAX_RETRANSMIT** times until the matching CoAP ACK or RST is received.

The Application Server MUST reply to CoAP CON messages as soon as possible.

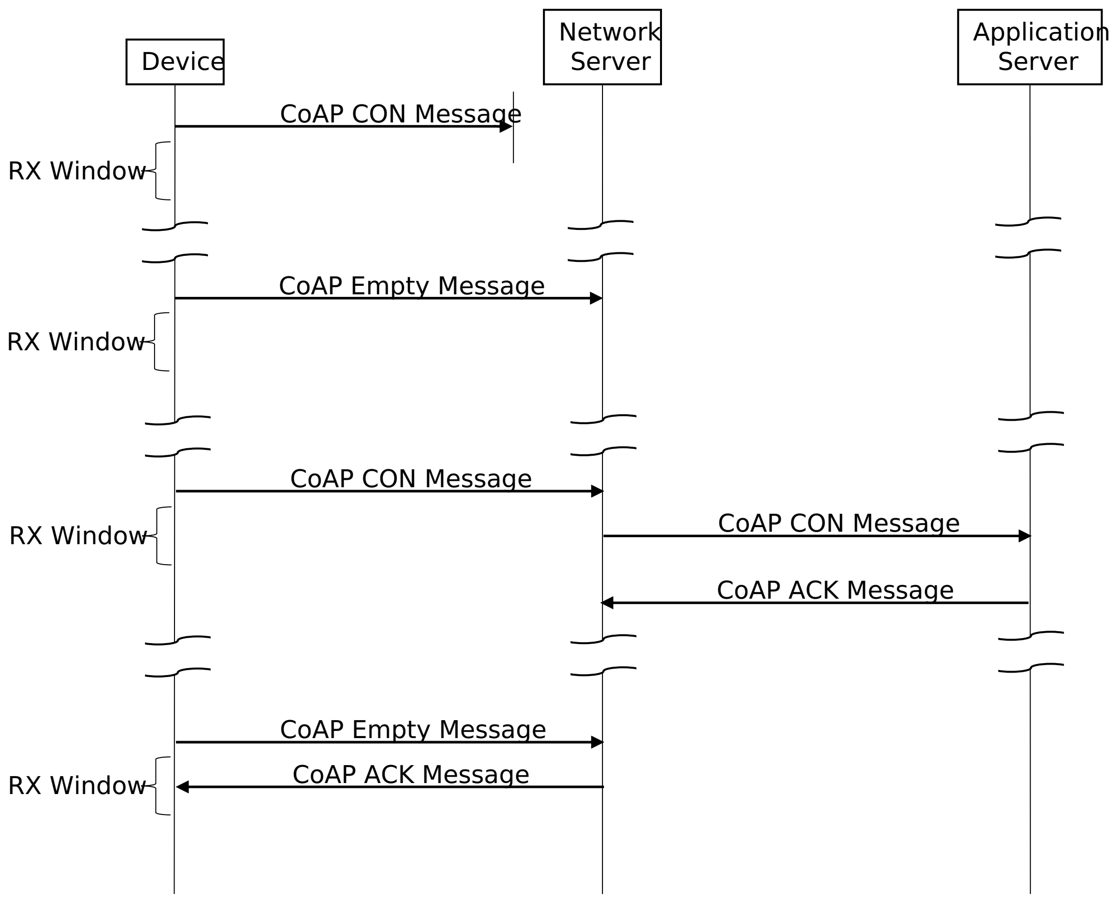

Figure: C.3.2.1-1 Example of CoAP request from Endpoint using LoRaWAN unconfirmable messages

C.3.2.2 Requests originating from the Application Server

1. The Application Server sends a CoAP CON message.
2. When the Network Server notifies the Application Server that the CoAP message was sent to the Endpoint, the Application Server waits for the CoAP ACK or RST message from the Endpoint during **ACK_TIMEOUT** seconds.
3. If the CoAP ACK or RST message was not received after **ACK_TIMEOUT** seconds, the Application Server resends the

initial CoAP CON message.

The Application Server repeats steps 2. and 3. MAX_RETRANSMIT times until the matching CoAP ACK or RST is received.

When receiving a CoAP CON message in an unconfirmable LoRaWAN packet, the Endpoint MUST reply in the ACK_TIMEOUT timeframe.

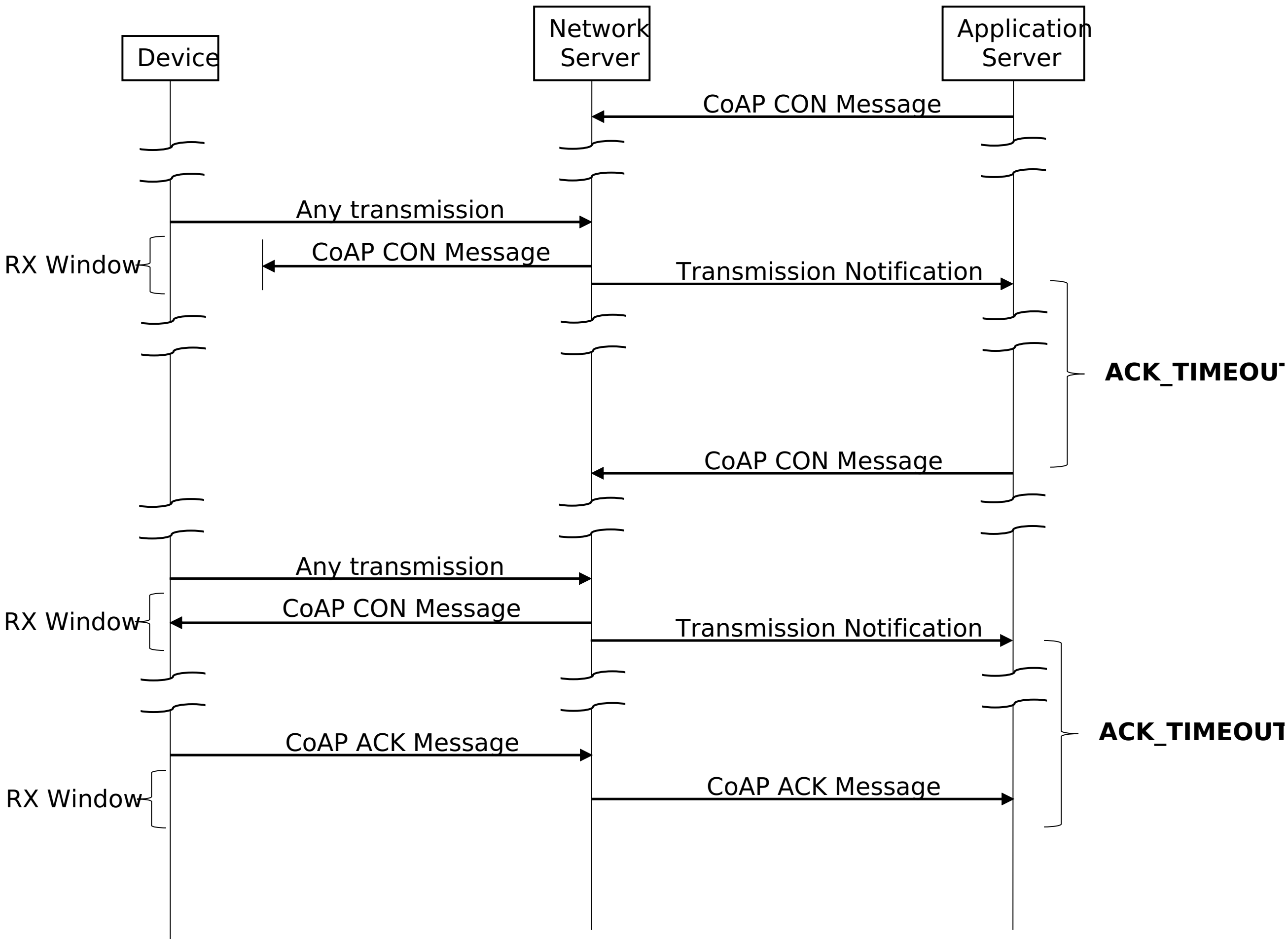

Figure: C.3.2.2-1 Example of CoAP request from Application Server using LoRaWAN unconfirmable messages

Appendix D. LwM2M over 3GPP CIoT – NB-IoT and LTE-M

D.1 Introduction

Note: The 3GPP vocabulary and abbreviations used in the Annex are explained in [3GPP-TR_21.905].

3GPP has specified Narrow-Band IoT (NB-IoT) and eMTC enhancements (aka. LTE Cat M1), as part of their Release 13. NB-IoT and CAT-M1 includes solutions for support of infrequent data transmission via user plane and via control plane. The user plane solution includes IP data and SMS support. The control plane solution includes IP data, non-IP data and SMS support. The main focus of this annex is on considerations and current limitations when running CoAP over the non-IP mode. Figure: D.1-1 3GPP CIoT architecture and the LwM2M protocol stack shows the 3GPP CIoT architecture, as described in [3GPP-TR_23.720] and the LwM2M protocol stack. The C-SGN combines the functionality of the MME, S-GW, and P-GW.

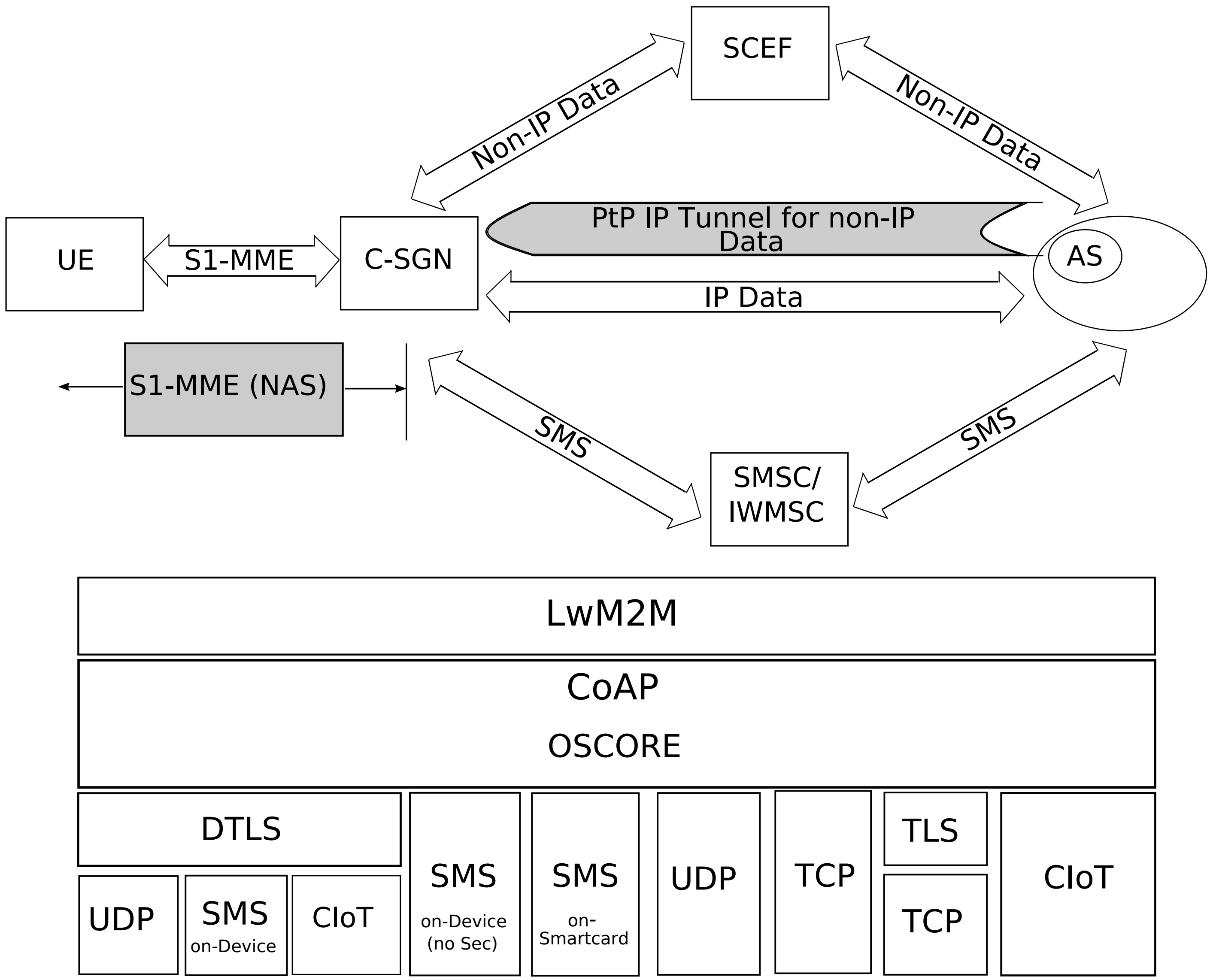

Figure: D.1-1 3GPP CIoT architecture and the LwM2M protocol stack

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 non-IP mode some limitations and considerations apply which are explained in the following sub-sections.

D.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).

D.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). [3GPP-TS_29.122] defines the T8 Reference point between the SCEF and the SCS/AS and specifies the RESTful APIs that allow the SCS/AS to access the services and capabilities provided by 3GPP network entities and securely exposed by the SCEF.

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]).

D.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_24.301] clause 6.6.4.2, 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 (urn:oma:lwm2m:oma:17).

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, else it may result in multiple retransmissions in the lower layers effectively undermining the expected results or performance.

The PCO is also used to covey 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 Serving PLMN Rate Control Resource in the Cellular Network Connectivity Object (urn:oma:lwm2m:oma:10) or via the APN Rate Control Resource in the APN Connection Profile Object (urn:oma:lwm2m:oma:11).

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.

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 am 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: D.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.

D.5 Large data transport with NB-IoT/CAT-M1

Even NB-IoT/CAT-M1 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, in which case these large data transport may not be feasible directly.

IETF has defined segmentation handling at the CoAP layer for large file transfer e.g. firmware updates. Blockwise transfers in [CoAP_BLOCKWISE] and [LwM2M-CORE] captures the details in LwM2M Object Firmware Update (ID:5)

CoAP block transfer MAY be used with for carrying CoAP over NB-IoT/CAT-M1. This option can be used with IP-mode and non-IP-mode.

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

D.6 Message buffering

NB-IoT/CAT-M1 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 not reachable, like device in sleeping mode or for other reasons like applied mobile network controls, downlink messages needs 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, this 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 the case that LwM2M queue mode is planned to be used, “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/CAT-M1 implementations need to support also time critical use cases e.g. fire alarms.

D.7 NB-IoT/CAT-M1 transport configuration options

Various configuration options for NB-IoT transport are provided via the ConnMgmt enabler.

Editor's note: Reference to be added

D.8 Timer considerations

D.8.1 Introduction

3GPP Rel-13 NB-IoT/CAT-M1 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.

D.8.2 3GPP Parameters

Parameter	Range	Purpose/how it's used by the device
PSM Timer, Extended T3412	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, T3324	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 T3324 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: D.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 on Resources which the device should request from the network. By observing this resource, the LwM2M also receives the finally applied timer in case the network has not accepted the device's request.

D.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: D.8.3-1 LwM2M Parameters

D.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 <= T3412, 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 T3412 < 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 T3412 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 and 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 and Pmax.

D.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.

A 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:

- Extended T3412 > LwM2M Registration lifetime > Pmax, 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.
- Pmin < eDRX < Pmax <= translated value of Rate control (in deci-hours) to time interval. The assumption is Pmin 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 Pmax) 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 Pmax requirements.
- When rate control is applied the Pmin and Pmax setting need to be chosen in a way to avoid notifications in a higher rate than rate control allows.