ETSI TS 102 921 V1.2.1 (2013-06)

Technical Specification

Machine-to-Machine communications (M2M);
mIa, dIa and mId interfaces

Reference

RTS/M2M-00010ed121

Keywords

interface, M2M, protocol, service

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Machine-to-Machine communications (M2M).

1 Scope

The present document contains the stage 3 specification for the mIa, dIa and mId reference points of the M2M architecture as identified in TS 102 690 [2].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ISO 8601: "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[2] ETSI TS 102 690: "Machine-to-Machine communications (M2M); Functional architecture".

[3] ETSI TS 102 671 (V9.0.0): "Smart Cards; Machine to Machine UICC; Physical and logical characteristics (Release 9)".

[4] ETSI TS 102 310 (V9.0.0): "Smart Cards; Extensible Authentication Protocol support in the UICC (Release 9)".

[5] ETSI TS 124 109: "Universal Mobile Telecommunications System (UMTS); LTE; Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details (3GPP TS 24.109)".

[6] ETSI TS 133 220: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) (3GPP TS 33.220)".

[7] ETSI TS 133 222: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (3GPP TS 33.222)".

[8] ETSI TS 129 109: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Stage 3 (3GPP TS 29.109)".

[9] OMA-TS-REST-NetAPI-Common-V1-0: "Common definitions for RESTful Network APIs".

NOTE: Available at http://www.openmobilealliance.org/.

[10] Recommendation ITU-T X.891/ISO/IEC 24824-1: "Information technology -- Generic applications of ASN.1: Fast infoset".

[11] W3C Recommendation 28 October 2004: "XML Schema Part 2: Datatypes Second Edition".

NOTE: Available at http://www.w3.org/TR/xmlschema-2/.

[12] W3C Recommendation 26 November 2008: "Extensible Markup Language (XML) 1.0 (Fifth Edition)".

NOTE: Available at http://www.w3.org/TR/REC-xml/.

[13] W3C Recommendation 10 March 2011: "Efficient XML Interchange (EXI) Format 1.0".

NOTE: Available at http://www.w3.org/TR/exi/.

[14] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[15] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[16] IETF RFC 2047: "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text".

[17] IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

[18] IETF RFC 2617: "HTTP Authentication: Basic and Digest Access Authentication".

[19] Void.

[20] IETF RFC 3447: "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1".

[21] Void.

[22] IETF RFC 3748: "Extensible Authentication Protocol (EAP)".

[23] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[24] IETF RFC 4055: "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[25] IETF RFC 4186: "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)".

[26] IETF RFC 4187: "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)".

[27] IETF RFC 4279: "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)".

[28] IETF RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1".

[29] Void.

[30] IETF RFC 4366: "Transport Layer Security (TLS) Extensions".

[31] IETF RFC 4627: "The application/json Media Type for JavaScript Object Notation (JSON)".

[32] IETF RFC 5091: "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems".

[33] IETF RFC 5191: "Protocol for Carrying Authentication for Network Access (PANA)".

[34] IETF RFC 5216: "The EAP-TLS Authentication Protocol".

[35] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[36] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[37] IETF RFC 5408: "Identity-Based Encryption Architecture and Supporting Data Structures".

[38] IETF RFC 5448: "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')".

[39] IETF RFC 5433: "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method".

[40] IETF RFC 5487: "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode".

[41] IETF RFC 6066: "Transport Layer Security (TLS) Extensions: Extension Definitions".

[42] IETF RFC 6267: "MIKEY-IBAKE: Identity-Based Authenticated Key Exchange (IBAKE) Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)".

[43] FIPS 180-2 (August 2002, with Change Notice 1, February 2004): "Secure Hash Standard".

[44] ISO/IEC/IEEE 9945 (2009): "Information technology -- Portable Operating System Interface (POSIX®) Base Specifications, Issue 7".

[45] IETF draft-ietf-core-coap-17: "Constrained Application Protocol (CoAP)".

NOTE: Available at http://tools.ietf.org/html/draft-ietf-core-coap-17.

[46] IETF draft-ietf-core-block-11: "Blockwise transfers in CoAP".

NOTE: Available at http://tools.ietf.org/html/draft-ietf-core-block-11.

[47] IETF draft-ietf-core-observe-08: "Observing Resources in CoAP".

NOTE: Available at http://tools.ietf.org/html/draft-ietf-core-observe-08.

[48] IETF draft-cakulec-emu-eap-ibake-01: "An EAP Authentication Method Based on Identity-Based Authenticated Key Exchange".

NOTE: Expired Internet Draft, see https://datatracker.ietf.org/doc/draft-cakulev-emu-eap-ibake/.

[49] TR-069 (Issue 1 - Amendement 4): "CPE WAN Management Protocol".

NOTE: Available at http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf.

[50] OMA-TS-DM-StdObj-V1-3 (Version 1.3): "Device Management Standardization Objects".

[51] OMA-TS-DCMO-V1-0 (Version 1.0): "Device Capability Management Object".

[52] OMA-TS-DiagMon-Functions-V1-0 (Version 1.0): "DiagMon Functions Supplemental Specification".

[53] OMA-TS-LAWMO-V1-0 (Version 1.0): "Lock and Wipe Management Object".

[54] OMA-TS-DiagMonTrapMOFrame-V1-2 (Version 1.2): "Diagnostics and Monitoring Trap Framework Management Object".

[55] OMA-TS-DiagMonTrapEvents-V1-2 (Version 1.2): "Diagnostics and Monitoring Trap Events Specifications".

[56] OMA-TS-DM-FUMO-V1-0 (Version 1.0): "Firmware Update Management Object".

[57] OMA-TS-DM-SCOMO-V1-0-20090903-C: "Software Component Management Object, version 1.0".

[58] Broadband Forum TR-106 (Issue 1): "Data Model Template for TR-069-Enabled Devices, Amendment 6".

[59] Broadband Forum TR-157 (Issue 1): "Component Objects for CWMP, Amendment 5, November 2011".

[60] Broadband Forum TR-181 (Issue 2): "Device Data Model for TR-069, Amendment 4".

[61] W3C Recommendation 10 June 2008: "XML Signature Syntax and Processing (Second Edition)".

NOTE: Available at http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.

[62] Void.

[63] W3C Recommendation 25 January 2005: "XML-binary Optimized Packaging".

NOTE: Available at http://www.w3.org/TR/2005/REC-xop10-20050125/.

[64] W3C Recommendation 25 January 2005: "SOAP Message Transmission Optimization Mechanism".

NOTE: Available at http://www.w3.org/TR/soap12-mtom/.

[65] OMA-TS-MLP-V3-3-20110719-A (Version 3.3): "Mobile Location Protocol".

[66] OMA-TS-DM-Protocol-V1-3: "Device Management Protocol".

[67] OMA-TS-DM-Notification-V1-3-20101207-C: "OMA Device Management Notification Initiated Session, version 1.3".

[68] OMA-TS-DM-RepPro-V1-3-20101207-C: "OMA Device Management Representation Protocol, version 1.3".

[69] IETF RFC 5705: "Keying Material Exporters for Transport Layer Security (TLS)".

[70] IETF RFC 3394: "Advanced Encryption Standard (AES) Key Wrap Algorithm".

[71] IETF RFC 5489: "ECDHE-PSK Cipher Suites for Transport Layer Security (TLS)".

[72] IETF RFC 6347: "Datagram Transport Layer Security Version 1.2".

[73] W3C Candidate Recommendation 13 March 2012: "XML Encryption Syntax and Processing".

NOTE: Available at http://www.w3.org/TR/2012/CR-xmlenc-core1-20120313.

[74] IETF draft-ietf-httpbis-p6-cache-22: "Hypertext Transfer Protocol (HTTP/1.1): Caching".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-httpbis-p6-cache/

[75] IEEE 802.1X: "IEEE Standard for local and metropolitan area networks - Port-Based Network Access Control".

[76] IEEE 802.16: "IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems".

[77] IETF RFC 5872: "IANA Rules for the Protocol for Carrying Authentication for Network Access (PANA)".

[78] IETF RFC 2865: "Remote Authentication Dial In User Service (RADIUS)".

[79] IETF RFC 6733: "Diameter Base Protocol".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 102 725: "Machine-to-Machine communications (M2M); Definitions".

[i.2] IETF draft-yegin-coap-security-options-00: "CoAP Security Options".

NOTE: Status is expired, but still available at https://datatracker.ietf.org/doc/draft-yegin-coap-security-options/.

[i.3] IETF RFC 6202: "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP".

[i.4] IETF RFC 4825: "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)".

[i.5]	Void.
[i.6]	IETF RFC 3588: "Diameter Base Protocol".

3 Definitions, symbols, abbreviations and conventions

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 102 725 [i.1] apply.

3.2 Symbols

For the purposes of the present document, the symbols given in TR 102 725 [i.1] and the following apply:

|	Concatenation

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 102 725 [i.1] apply.

3.4 Conventions

All data variables in the present document are presented with the most significant substring on the left hand side and the least significant substring on the right hand side. A substring may be a bit, byte or other arbitrary length bitstring. The symbols " in a formula are used to indicate a null-terminated constant octet string (ASCII-encoded). The symbols " are not part of the delimited constant string.

4 Overview

The M2M system is resource based and the API uses the RESTful approach.

Operation on resource among Applications and SCLs and between SCL instances are supported by means of Methods, Methods constitute the communication on the mIa, dIa and mId APIs. Each method conveys a set of information defined as Method Attributes.

Three main aspects are specified on the reference points in order to provide equivalent full standardized interfaces:

- The protocol/API definition.
- Definition of Resources and Sub-Resources using the APIs.
- The interaction with/reuse of existing protocols, including the identification of key underlying protocols.

Details of security procedures are addressed in the present document.

In agreement with M2M architectural principle:

- The interfaces aim to be applicable to a wide range of network technology, so when specific potential bindings are defined, these bindings are not limiting the applicability when used in other networks (e.g. using different protocols).
- The interfaces are applicable to all the M2M application domains, the interfaces aim to be application and access independent.

A Resource is a uniquely addressable entity in the RESTful vocabulary. Each resource has a representation that shall be transferred and manipulated with the verbs. A resource shall be addressed using a Universal Resource Identifier (URI).

The present document contains:

- Primitive descriptions.
- Data type definitions.
- Mapping to HTTP and CoAP.
- XML definitions of the resources.
- Security procedures.

5 General security aspects

5.1 Key provisioning and hierarchy derivation

The ETSI M2M key hierarchy is defined in TS 102 690 [2], clause 8. The following clauses describe the derivation procedures of the hierarchy.

5.1.1 Kmr provisioning

The M2M Service Root key (Kmr) shall have a length equal to 256 bits.

Kmr may be provisioned offline (i.e. prior to the initial communication between D/G M2M Node and Network M2M Node or MSBF) into D/G M2M Node or UICC card. Kmr may be provisioned online (i.e. during the initial communication between D/G M2M Node and Network M2M Node or MSBF) based on one of the following bootstrapping procedures.

5.1.1.1 Kmr provisioning independent of access network credentials

If IBAKE based bootstrapping carried using EAP over PANA is applied, then Kmr shall be provisioned as per the procedure described in clause 6.3.1.2. If TLS based bootstrapping carried using EAP over PANA is applied, then Kmr shall be provisioned as per the procedure described in clause 6.3.1.3.

If TLS based bootstrapping carried using TCP is applied, then Kmr shall be provisioned as per the procedure described in clause 6.3.2.

While other bootstrapping methods that are independent of access network credentials can be applicable, they are not included in the present document.

5.1.1.2 Kmr provisioning based on access network credentials

If GBA based bootstrapping is applied, then Kmr shall be provisioned as per the procedure described in clause 6.2.1.

If EAP based bootstrapping using SIM/AKA access network credentials is applied, then Kmr shall be provisioned as per the procedure described in clause 6.2.2.

If bootstrapping using EAP-based network access authentication is applied, then Kmr shall be provisioned as per the procedure described in clause 6.2.3.

5.1.1.3 Kmr refresh and invalidation

Kmr may be refreshed via a new manual pre-provisioning or via the invocation of a new bootstrapping procedure using either of the procedures listed in clauses 5.1.1.1 and 5.1.1.2, and the referenced clauses therein. Kmr may be invalidated based on M2M Service Provider policy.

5.1.2 Kmc derivation

The M2M Service Connection key (Kmc) shall have a length equal to 256 bits.

Kmc shall be associated to a specific M2M Service Connection and shall be derived only upon successful mutual authentication between the D/G M2M Node and the Network M2M Node, using either of the methods described below.

Kmc may be used for securing sensitive information exchanged between D/GSCL and NSCL over mId in the context of channel security, where it is used to derive a PSK for TLS-PSK. Kmc may alternatively be used within the context of object security.

5.1.2.1 Kmc derivation in the case of EAP based mutual authentication and key agreement

If EAP over PANA is used for mutual authentication and key agreement using Network M2M Node, then the MSK produced by EAP shall be securely sent from the MAS to the M2M Network Node. Moreover in this case D/G M2M Node and Network M2M Node shall derive Kmc based on the formula described in clause 7.3 for EAP based mutual authentication and key agreement.

5.1.2.2 Kmc derivation in the case of GBA based mutual authentication and key agreement

If GBA is used for mutual authentication and key agreement, then D/G M2M Node and Network M2M Node shall agree on Kmc as described in clause 7.2.1 for GBA based mutual authentication and key agreement.

5.1.2.3 Kmc derivation in the case of TLS based mutual authentication and key agreement

If TLS-PSK is used for mutual authentication and key agreement, then Kmc shall be produced by MAS and shall be securely sent to Network M2M Node and to D/G M2M Node as per clause 7.4 for TLS based mutual authentication and key agreement.

5.1.2.4 Kmc refresh and invalidation

The Kmc key shall only be refreshed with every new mutual authentication and key agreement procedure. Kmc may be invalidated based on M2M Service Provider policy. Kmc shall be invalidated upon invalidation of the parent Kmr key.

5.2 Security Assumptions

In the following text, entity A has "authenticated" entity B if entity A has a degree of confidence (appropriate to the sensitivity of the resources that can be acted on by messages from entity) that messages claiming to be from entity B were sent by entity B and messages sent to entity B are observed only by entity B.

The M2M Security framework relies on following assumptions:

- An SCL and Applications local to that SCL shall authenticate each other (according to the above definition of "authenticated"). The present document does not include methods for authentication of an SCL and Applications local to that SCL; however some examples are provided in the note below.

- The Hosting SCL and Requesting SCL shall be mutually authenticated (according to the above definition of "mutually authenticated"). Mutual authentication of SCLs can be achieved by relying on access network layer security or by performing an M2M Service Connection Procedure and then applying the negotiated mId Security Procedure.

The following bullets list some example methods that can be used for mutual authentication of an SCL and an Application local to that SCL. This list is not exhaustive. The implementer is responsible for determining which method or methods are applicable. The present documentdoes not provide any details of these methods.

- The mutual authentication can use strong cryptographic processes.

- The mutual authentication can rely on passwords.

- Mutual authentication can be implicit in some scenarios. Some such scenarios are listed below - this list is not exhaustive and many additional such scenarios exist:
  - Example 1. In some scenarios, any application loaded onto the M2M Device/Gateway by the user may be implicitly trusted by the local SCL.
  - Example 2. In some scenarios, an M2M Device/Gateway is pre-provisioned with Applications that are trusted and the M2M Device/Gateway does not allow addition applications to be loaded.

6 M2M Service Bootstrapping

6.1 General Principles

TS 102 690 [2] provides a variety of options for M2M service bootstrapping, some of which rely on mechanisms that are not included in of the present specification. Therefore, the M2M bootstrap procedures described in this clause are all optional to implement and use on both the M2M core side and the M2M Device/Gateway side.

The bootstrapping procedures described in clause 6.2 assume the existence of a business relationship between the M2M Service Provider and the Access Network Provider. Any of the procedures described in clauses 6.2.1 (GBA based), 6.2.2 (SIM/AKA based) or 6.2.3 (other EAP/PANA based) may be implemented in the M2M Core side and in the M2M device/gateway side, in compliance with the specifications of the corresponding subclauses.

The bootstrapping procedures specified in clause 6.3 are independent on the access network. Any of the procedures described in clauses 6.3.1 (EAP/PANA based) or 6.3.2 (TLS/TCP based) may be implemented in the M2M core side and in the M2M device/gateway side, in compliance with the specifications of the corresponding clauses.

NOTE: Upon each failure of a M2M Service Bootstrap Procedure, the D/G M2M Node is recommended to increase the time that it waits before attempting an M2M Service Bootstrap Procedure again. This time can be reset to its initial value upon successful M2M Service Bootstrap Procedure.

Procedures using TLS, whether based on EAP/PANA (clause 6.3.1.3) or TLS/TCP (clause 6.3.2) shall comply with the specifications for TLS/Certificate-based M2M Service bootstrap procedures of clause 6.3.3.

Additionally, procedures using the M2M Service bootstrap parameter delivery procedures using HTTP, whether based on GBA, clause 6.2.1 or on TLS/TCP, clause 6.3.2, shall comply with the specifications of clause 6.4.

6.2 Access Network Assisted M2M Service Bootstrap Procedure

6.2.1 GBA-based M2M Service Bootstrap Procedure

As described in clause 8.3.2.1 of TS 102 690 [2], the GBA-based M2M service bootstrap shall start with the D/G M2M Node and the BSF carrying out GBA bootstrapping over the Ub interface. There are two modes of GBA: ME-based (GBA_ME) and UICC-based (GBA_U). The latter requires that the UICC is GBA aware. The BSF shall decide which mode to run based on the UICC capability indicated in the GBA user security settings (GUSS). TS 124 109 [5] defines the details for the HTTP Digest AKA based implementation of the Ub interface.

After a successful GBA bootstrapping, the D/G M2M Node and the BSF share a security association which consists of a bootstrapping transaction identifier (B-TID) and key material (GBA Ks).

6.2.1.1 Optional use of GBA_U with Ks_int_NAF

If the HTTP client is implemented in the UICC, Ks_int_NAF shall be used as Kmr. This option is recommended because if Ks_ext_NAF or Ks_NAF are used as a password in the HTTP digest calculation (see clause 6.2.1.2), then they could be exposed to malicious applications in the device environment at any time, while Ks_int_NAF remains protected inside the UICC.

6.2.1.2 HTTP Digest Authentication and bootstrap parameter delivery

Figure 6.1 shows the HTTP Digest Authentication procedure with the following sub-steps:

Step 2.1

The D/G M2M Node shall send an HTTP POST request to the MSBF/NAF. It shall indicate to the MSBF/NAF that it supports 3GPP-bootstrapping based HTTP Digest authentication by including a "product" token to the "User-Agent" header that is a static string, either "3gpp-gba-uicc" if the HTTP client application resides in the UICC (see clause 6.2.1.1) or "3gpp-gba" if the HTTP client application resides in the ME. The Request-URI shall be "/" and the "Host" header shall contain the FQDN of the MSBF/NAF.

Step 2.2

The MSBF/NAF shall respond with HTTP response code 401 "Unauthorized" which shall contain a "WWW-Authenticate" header. In the "WWW-Authenticate" header, the "realm" attribute shall contain two parts delimited by "@" sign. The first part is the constant string "3GPP-bootstrapping-uicc" if Ks_int_NAF is used as Kmr (see clause 6.2.1.1) or "3GPP-bootstrapping" otherwise, and the second part shall be the FQDN of the MSBF/NAF (e.g. "3GPP-bootstrapping@msbf.operator.com"). The "qop" attribute shall be set to "auth-int" meaning that the payload of the following HTTP requests and responses are to be both authenticated and integrity protected.

Step 2.3

Upon reception of the HTTP response from the MSBF/NAF, the D/G M2M Node verifies that the second part of the "realm" attribute is equal to the FQDN of the MSBF/NAF. If not, the D/G Node may repeat Step 2.1. Otherwise, the D/G M2M Node shall perform Step 1-2 of the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4.5. The =bootstrapParamSetExecuteRequestIndication primitive shall be encoded into an HTTP request according to the HTTP mapping specified in annex C.

The resulting HTTP request shall carry data needed for HTTP Digest Authentication. In the "Authorization" header, the "username" attribute is the bootstrapping transaction identifier B-TID. The password used in the digest calculation shall be Ks_NAF in the case of GBA_ME and Ks_int/ext_ NAF (see clause 6.2.1.1) in the case of GBA_U, Base64 encoded. See clause 6.2.1.3 for the calculation of Ks_NAF and Ks_int/ext_NAF.

The D/G M2M Node shall then perform Step 3-4 of the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4.5.

Step 2.4

Upon reception of the HTTP request from the D/G M2M Node, the MSBF/NAF extracts B-TID from the "username" attribute and uses it to retrieve Ks_NAF or Ks_int/ext_NAF and associated key lifetime from the BSF. The retrieval is done over the Zn interface as specified in TS 129 109 [8]. The MSBF/NAF verifies the "Authorization" header by calculating the corresponding digest value using Ks_NAF or Ks_int/ext_NAF and comparing the calculated value with the received value in the header. It shall also verify that the FQDN in the "realm" attribute matches its own. The MSBF/NAF shall perform Step 5-8 of the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4.5. The BootstrapParamSetExecuteResponseConfirm primitive shall be encoded into an HTTP response according to the HTTP mapping specified in annex C. The "Authentication-Info" header in this response shall include the "rspauth" attribute to carry response digest. The MSBF/NAF shall then perform Step 9 of the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4.5.

Upon reception of the HTTP response from the MSBF/NAF, the D/G M2M Node verifies the "Authentication-Info" header. It shall then perform Step 10-11 of the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4.5.

Annex B of TS 124 109 [5] describes signalling flows for HTTP Digest Authentication over the Ua interface.

Authentication failures shall be handled as they are described in RFC 2617 [18].

6.2.1.3 M2M Root Key (Kmr) derivation

If GBA_ME is run, Ks_NAF as derived by the D/G M2M Node and the BSF shall be used as the M2M Root key (Kmr):

- Ks_NAF = KDF (Ks, "gba-me", RAND, IMPI, NAF_Id).

If GBA_U is run, two NAF-specific keys are derived:

- Ks_ext_NAF = KDF(Ks, "gba-me", RAND, IMPI, NAF_Id).
- Ks_int_NAF = KDF(Ks, "gba-u", RAND, IMPI, NAF_Id).
- If the HTTP client application resides in the UICC, then Ks_int_NAF shall be used as Kmr. Otherwise, Ks_ext_NAF shall be used as Kmr.

Here, KDF is the key derivation function as specified in annex B of TS 133 220 [6], and the key derivation parameters include the RAND (a part of the authentication vector), IMPI (for USIM, the IMPI is derived from IMSI), and the NAF_Id. The derived key is 256 bit long.

The NAF_Id is constructed as follows: NAF_Id = FQDN of the NAF | Ua security protocol identifier. As MSBF is the NAF, the FQDN of the MSBF shall be used. The Ua security protocol identifier is specified in annex H of TS 133 220 [6]. As HTTP digest authentication according to TS 124 109 [5] is used here as the Ua security protocol, the identifier shall be (0x01,0x00,0x00,0x00,0x02).

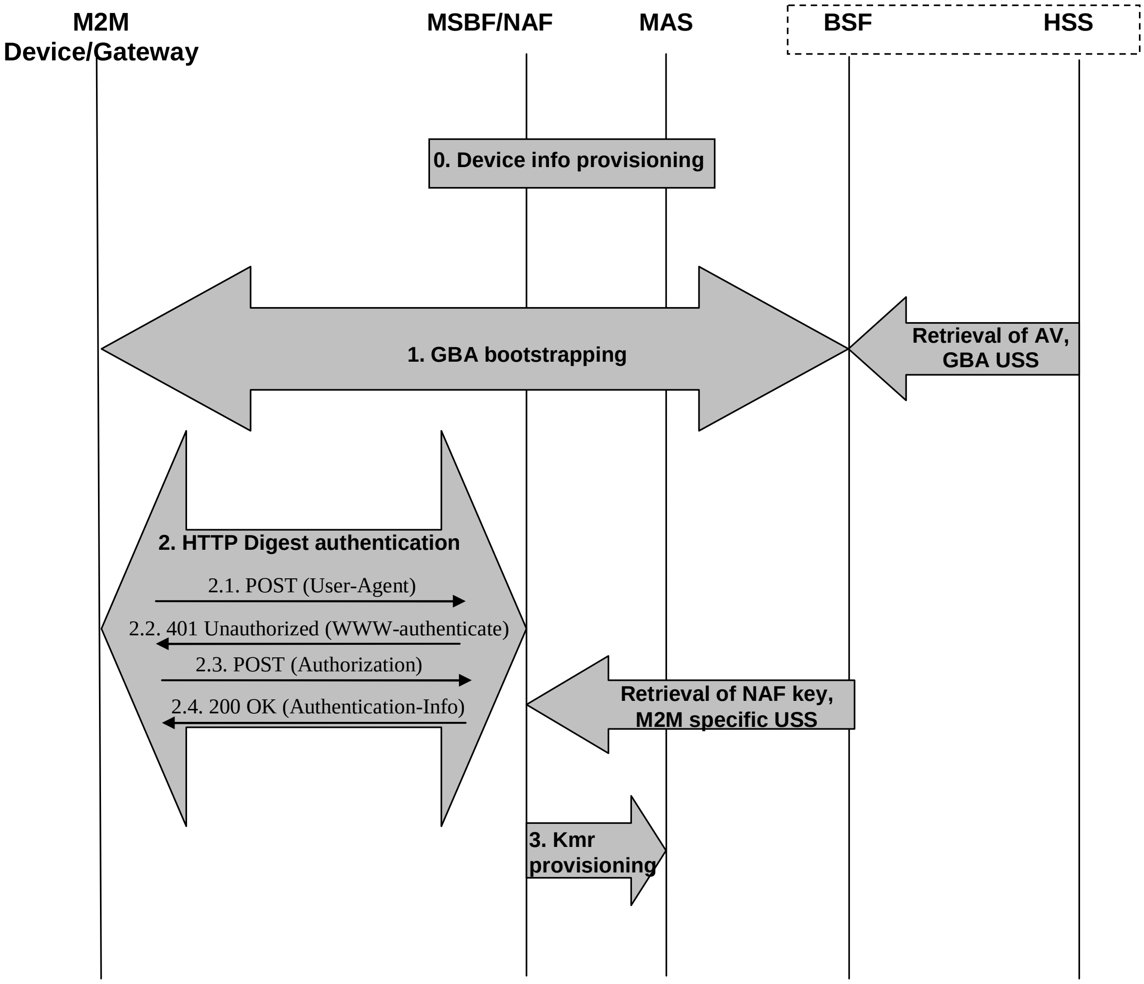

M2M Device/Gateway
MSBF/NAF
MAS
BSF
HSS
0. Device info provisioning
1. GBA bootstrapping
Retrieval of AV, GBA USS
2. HTTP Digest authentication
2.1. POST (User-Agent)
2.2. 401 Unauthorized (WWW-authenticate)
2.3. POST (Authorization)
2.4. 200 OK (Authentication-Info)
Retrieval of NAF key, M2M specific USS
3. Kmr provisioning

Figure 6.1: M2M Service Bootstrap based on GBA

6.2.2 EAP-based bootstrapping procedure using SIM/AKA Access Network Credentials

See TS 102 690 [2] for details.

6.2.3 Bootstrapping from EAP-based access network layer

This procedure is based on utilizing the network access authentication procedure for bootstrapping the D/G M2M Node with the M2M service. Instead of authenticating the D/G M2M Node twice (once for the network access service, and another for the M2M service), the D/G M2M Node is authenticated once during the network access procedure and the cryptographic material produced during that procedure is used for generating the Kmr.

This procedure shall only be used with networks that are already using EAP-based mutual authentication and key agreement for network access (e.g. WiFi, Ethernet, WiMAX, Zigbee, etc.). Furthermore, there shall be a relationship between the network access provider and the M2M service provider, so that the network access provider and the M2M service provider can share keying material among themselves. Such relationships include but are not limited to the network access provider and the M2M service provider being the same operator.

Figure 6.2 depicts the generic call flow for this procedure.

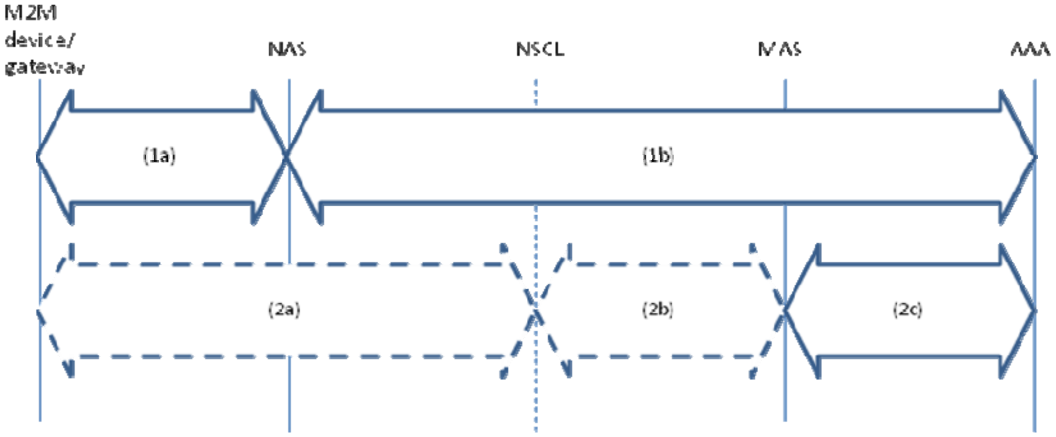


Figure 6.2: Bootstrapping via network access authentication

In figure 6.2, NAS represents the Network Access Server, and AAA represents the Authentication, Authorization, and Accounting Server. These two network elements are used for network access authentication procedure.

The AAA and the MAS are shown separately. The AAA and the MAS may be co-located depending on the deployment.

NOTE: Since the AAA is responsible for generating and delivering the Kmr to the MAS, the AAA virtually assumes the role of MSBF in this procedure.

Steps 1a and 1b are part of the network access authentication procedure, which is outside the scope of the present document.

Steps 2a and 2b are part of the M2M Service Connection procedure which acts as a trigger for the MAS to obtain the Kmr from the AAA. Therefore, these two steps and the associated network element Network M2M Node are shown with dotted-lines.

Step 1. EAP-based network access authentication.

In this step, EAP shall be used for network access authentication among the D/G M2M Node, NAS, and AAA. The choice of EAP lower layers (e.g. IEEE 802.1X [75], PANA [77], IEEE 802.16 PKMv2 [76], RADIUS [78], Diameter [79], etc.) and the EAP authentication methods depend on the access network architecture/deployment. Such details have no impact on this procedure and therefore they are outside the scope of the present document.

The EAP authentication produces EMSK (Extended Master Session Key - RFC 3748 [22]) at the end of successful network access authentication. This key (EMSK) shall be used for generating the Kmr as described in the following step.

Step 2. D/G M2M Service Connection triggering the generation/delivery of Kmr.

The D/G M2M Node and the AAA shall use the following formula for generating the Kmr:

- Kmr = Hash(EMSK, "ETSI M2M Device-Network Root Key" |D/G M2M-Node-ID | M2M-SP-ID).

Where:

- Hash is HMAC-SHA256.
- EMSK is the Extended Master Session Key (RFC 3748 [22]) generated by the EAP authentication method. The EMSK shall be made available to the D/G M2M Node by the EAP peer implementation, and to the AAA by the EAP authentication server implementation.
- D/G M2M-Node-ID is the identifier assigned to the D/G M2M Node. If no such identifier is assigned, then the Pre-provisioned-ID shall be used (TS 102 690 [2]) (e.g. device MAC address). How that identifier is determined is specific to the access network type, and therefore the determination of this identifier is outside the scope of the present document.
- M2M-SP-ID is the identifier of the M2M Service Provider. This identifier may be same as the network access provider's identifier. How that identifier is determined is outside the scope of the present document.

The D/G M2M Node shall generate the Kmr before entering the M2M Service Connection procedure (Step 2a). The D/G M2M Node is not required to generate Kmr each time network access authentication procedure is executed. The D/G M2M Node shall use the freshest EMSK available at the time of generating Kmr.

When the Network M2M Node enters the M2M Service Connection Procedure (Step 2a), it shall contact the MAS for the authentication/authorization of the D/G M2M Node (Step 2b). The MAS shall request the Kmr from the AAA server (Step 2c) upon receipt of the request from the Network M2M Node. The AAA server shall generate and deliver the Kmr to the MAS upon receipt of such a request (Step 2c). The AAA server is not required to generate Kmr each time network access authentication procedure is executed. The AAA server shall use the freshest EMSK available at the time of generating Kmr.

The MAS-AAA interface is not included in of the present document. These two network elements may be co-located.

The MAS shall know the identity of the AAA for a given D/G M2M Node, so that the MAS knows where to obtain the Kmr from.

Either the MAS and the AAA server shall both use the same D/G M2M-Node-ID, or if the MAS and the AAA server use different identifiers (for the D/G M2M-Node-ID) on each side then the MAS and the AAA server shall be aware of the binding between the two identifiers.

6.3 Bootstrapping using other methods

6.3.1 Bootstrapping methods using EAP over PANA

6.3.1.1 Generic procedure

6.3.1.1.1 Bootstrapping

EAP/PANA can be used by the D/G M2M Node and the Network M2M Node for performing M2M Service Bootstrapping procedure. This procedure registers the D/G M2M Node with a Network M2M Node by performing end-point authentication/authorization, M2M bootstrapping related parameter discovery, and key agreement.

The following details apply to M2M networks using EAP/PANA for M2M Bootstrap Procedure.

EAP [22] shall be used for mutual authentication between the D/G M2M Node and the MSBF via the Network M2M Node. Variants of EAP mutually-authenticating and key generating method (e.g. EAP-SIM [25], EAP-AKA [26], EAP-AKA' [38], EAP-TLS [34] and EAP-IBAKE [48]) may be used for this. PANA [33] is used as the transport protocol for EAP and M2M bootstrapping parameters.

The D/G M2M Node shall implement the EAP peer functionality, the Network M2M Node shall implement the EAP authenticator functionality, and the MSBF shall implement the EAP authentication server functionality [22] for executing EAP. Furthermore, the D/G M2M Node shall implement PANA Client (PaC) functionality, and the Network M2M Node shall implement PANA Authentication Agent (PAA) functionality [33] for transporting EAP and other parameters over PANA. There is no protocol mandate for the interface between the Network M2M Node and the MSBF since that interface is not included in the present document. RADIUS [78] and Diameter [i.6] are examples of possible protocols that can be used over that interface.

This bootstrapping procedure shall use the base PANA protocol [33] with additional AVPs. Only the new AVPs are described in the present document (see clause 12.1). Unless stated otherwise base protocol behaviour and AVPs shall be used. For example, when the present document requires a PANA packet to include a newly-defined AVP, implementers shall take that AVP in addition to the base AVPs as required by the base protocol [33] for the given packet type.

Figure 6.3 depicts the generic call flow for this procedure.

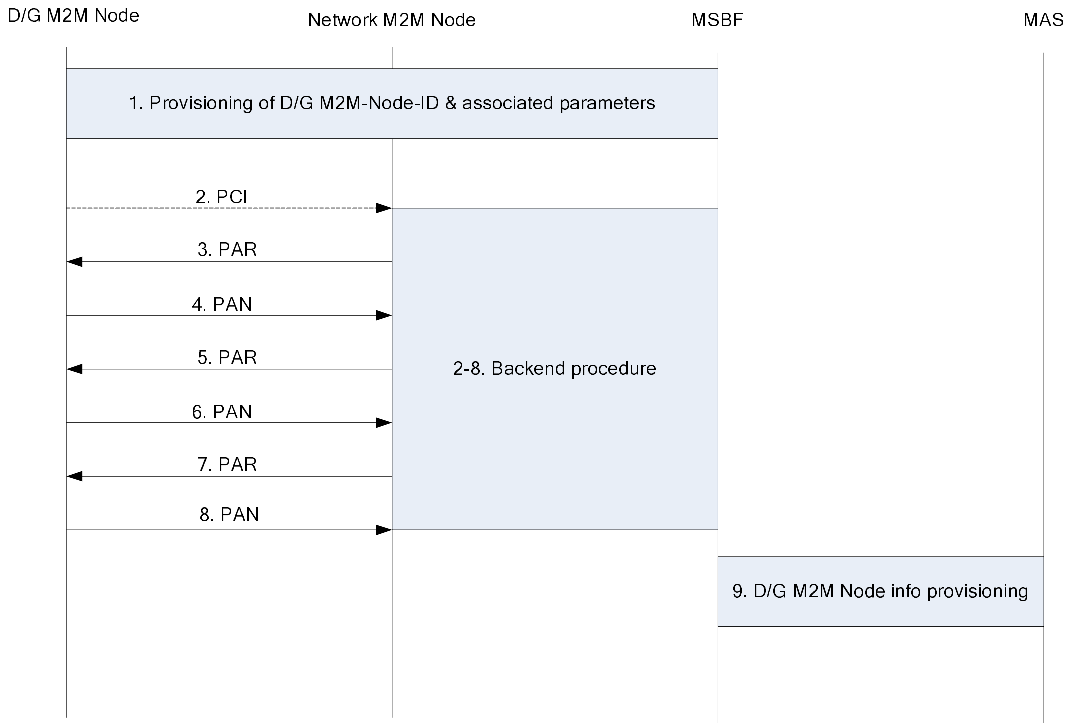


Figure 6.3: M2M Service Bootstrapping procedure using EAP/PANA

Step 1 between the D/G M2M Node and the MSBF and possibly the Network M2M Node, steps 2-8 between the Network M2M Node and the MSBF, and Step 9 between the MSBF and the MAS are not included in the present document. Therefore these steps are shown as block diagrams.

Step 1. Pre-provisioning.

The Pre-provisioned-ID of D/G M2M Node, as well as a set of other M2M Bootstrapping related parameters shall be provisioned to an MSBF, and also shall be pre-provisioned to the D/G M2M Node, prior to bootstrapping. Details of this step are outside the scope of the present document.

Step 2. D/G M2M Node initiating PANA.

The bootstrap procedure may be initiated either by the D/G M2M Node or the Network M2M Node. When the bootstrap procedure is initiated by the Network M2M Node, this step shall be omitted.

In scenarios where the bootstrap procedure is initiated by the D/G M2M Node, the D/G M2M Node shall send a PANA-Client-Initiation (PCI) packet to the Network M2M Node.

PCI shall include the following AVPs:

- M2M-Usage-Type carrying "M2M Bootstrapping"; and

- M2M-Node-ID carrying the device/gateway identifier.

PCI may also include the following AVPs:

- M2M-MSBF-ID carrying the identifier of MSBF;
- M2M-NSCL-ID carrying the identifier of NSCL;
- M2M-SP-ID carrying the identifier of the M2M Service Provider.

These optional AVPs shall be present when the D/G M2M Node intends to limit the procedure to such specific targets. For example, a D/G M2M Node can bootstrap with M2M-SP-ID = "M2M-Service-Provider.com" using M2M-MSBF-ID = "MSBF9.M2M-Service-Provider.com". For each of these types of AVP, if the AVP is not present in the PC, then this shall indicate that the D/G M2M Node is willing to bootstrap with any such element associated with that AVP.

The Network M2M Node shall ignore the incoming PCI if the PCI contains M2M-MSBF-ID, M2M-NSCL-ID, or M2M-SP-ID AVPs and the Network M2M Node is not configured to serve the MSBF or NSCL or M2M-SP identified in the AVPs.

Starting with Step 2, the Network M2M Node and MSBF may start signalling each other in coordination with the mId signalling. No details are provided about that part since the interface between Network M2M Node and MSBF is not included in the present document.

Steps 3. First PANA request packet from Network M2M Node.

The Network M2M Node shall send the first PANA-Authentication-Request (PAR) packet. Either this packet may be sent in response to a received PCI, or this packet may be an unsolicited packet.

- If the PAR is sent unsolicited, then in some cases the Network M2M Node knows the IP address of the D/G M2M Node, while in other cases the Network M2M Node does not know the IP address of the D/G M2M Node:
    - If the IP address is known to the Network M2M Node, the PAR shall be unicasted to that address. This PAR shall include the following AVPs: M2M-Usage-Type indicating "M2M Bootstrapping", M2M-MSBF-ID, M2M-NSCL-ID, M2M-SP-ID. The Network M2M Node is presumed to already know the values to assign to these AVPs.
    - If the IP address is not known to the Network M2M Node, then the PAR shall be either anycasted, multicasted, or broadcasted to an address where the D/G M2M Node is expected to be reachable. This PAR shall include the following AVPs: M2M-Usage-Type indicating "M2M Bootstrapping", M2M-MSBF-ID, M2M-NSCL-ID, M2M-SP-ID, M2M-Node-ID carrying the D/G M2M-Node-ID of the target D/G M2M Node.
- If the PAR is sent in response to the PCI, then it shall be unicasted to the IP address of the D/G M2M Node (in the source IP address of the IP packet containing the PCI). This PAR shall include the following AVPs: M2M-Usage-Type indicating "M2M Bootstrapping", M2M-MSBF-ID, M2M-NSCL-ID, M2M-SP-ID. The values carried in M2M-MSBF-ID, M2M-NSCL-ID, and M2M-SP-ID AVPs shall be copied from corresponding AVPs present in the PCI. The Network M2M Node shall choose values on its own for the AVP types that are not present in the PCI. A missing AVP of these types indicate the D/G M2M Node is willing to use any element (i.e. a wildcard value), as specified in Step 2.

The D/G M2M Node shall ignore an incoming PAR if that packet carries M2M-Node-ID AVP(s) and the D/G M2M Node's own D/G M2M-Node-ID does not match any one of the AVP values. This filtering allows a D/G M2M Node to avoid a PAR intended for other D/G M2M Nodes.

The D/G M2M Node may be configured to only use device-initiated bootstrapping for security reasons, in which case the D/G M2M Node shall ignore the unsolicited PAR.

The D/G M2M Node may ignore an incoming PAR if the PAR carries a value in one of the M2M-MSBF-ID, M2M-NSCL-ID, and M2M-SP-ID AVPs that does not match with the value sent in the corresponding AVP of the PCI.

The first EAP AVP may be included in this packet or the subsequent packet from Network M2M Node [33].

Step 4. D/G M2M Node responds to Network M2M Node.

The D/G M2M Node responds to the incoming PAR with a PANA-Authentication-Answer (PAN) packet, if that PAR is not ignored at Step 3. This PAN shall include the following AVP: M2M-Usage-Type indicating "M2M Bootstrapping".

Steps 5 and 6. Authentication.

These steps execute the EAP-based authentication by carrying the EAP protocol messages inside PANA EAP AVPs. One or more such round-trips is used depending on the selected EAP authentication method (see [22] and [33]).

Step 7. Final PAR from Network M2M Node.

This final PAR (Completion bit set) shall be sent from the Network M2M Node to signal the result of the bootstrapping procedure.

If the result indicates success, then the PAR shall include the following AVP:

- M2M-Bootstrap-Result indicating success.

The PAR may include the following AVPs:

1) M2M-Node-ID carrying the M2M Service Provider-assigned device/gateway identifier;
2) M2M-DSCL-ID carrying the D/GSCL-ID assigned to the Device/Gateway M2M Node by the network;
3) M2M-NSCL-ID carrying the NSCL-ID for the subsequent SCL procedures.

The M2M-Node-ID AVP shall be used if the M2M Service Provider prefers the Device/Gateway to use an identifier other than the pre-provisioned one.

Both the D/G M2M Node and the MSBF shall generate Kmr according to the following formula upon successful bootstrapping:

- Kmr = Hash(EMSK, "ETSI M2M Device-Network Root Key" | D/G M2M-Node-ID | M2M-SP-ID);

where:

- Hash is HMAC-SHA256.
- EMSK is the Extended Master Session Key [22] generated by the EAP authentication method. EMSK shall be made available to the D/G M2M Node by the EAP peer implementation, and to the MSBF by the EAP authentication server implementation.
- M2M-Node-ID is the value of the M2M-Node-ID AVP, if present in the final PAR. If that AVP is missing, the value of M2M-Node-ID in PCI/initial PAR shall be used instead.
- M2M-SP-ID is the value of the M2M-SP-ID AVP sent in the initial PAR.

The bootstrap procedure may be executed multiple times. Either a new PANA session may be established each time, or the new procedure may be executed within the already established PANA session (when available). In either case the newly-generated Kmr shall be distinguishable from the earlier instances. The following key index shall be used for this purpose:

- I_{Kmr} = Session-Identifier | Key-Id

where:

- Session-Identifier is the PANA session identifier assigned by the Network M2M Node.
- Key-Id is the PANA key identifier assigned by the Network M2M Node.

The lifetime of Kmr shall be set to the PANA Session-Lifetime assigned by the Network M2M Node.

Both the lifetime and the index of Kmr shall be stored along with the key itself. The procedure for passing these values to the MAS is not included in the present document.

If the result indicates failure (e.g. due to EAP authentication or authorization failure), then the PAR shall include the following AVP: M2M-Bootstrap-Result indicating failure.

Step 8. Final PAN from D/G M2M Node.

The D/G M2M Node shall send a PAN in response to the PAR.

Step 9. MSBF provisioning MAS.

The MSBF shall send the Kmr and D/G M2M-Node-ID to the MAS. The interface between the MSBF and the MAS are not included in the present document.

6.3.1.1.2 Bootstrap-Erase

The Erase procedure may be used when either the D/G M2M Node or the network desires to terminate the M2M service provided to the D/G M2M Node by the M2M Service Provider. A Bootstrap procedure followed by an erase procedure puts the D/G M2M Node back to its state prior to the bootstrapping with respect to the given M2M Service Provider. A D/G M2M Node in that state needs to re-execute bootstrapping if the D/G M2M Node desires to use M2M service of the given M2M Service Provider again.

The PANA Termination Procedure [33] shall be used for this purpose. A special payload (M2M-Erase-Token) shall signify this termination as Bootstrap-Erase Procedure. Either the D/G M2M Node or the network may initiate the procedure. On the network side, either the MAS or the MSBF may initiate the procedure. Whether the MAS initiates the process or the MSBF initiates the procedure is an operator policy and is not specified. Since the interface between the Network M2M Node and the MSBF/MAS is not included in the present document, details of how the MSBF or the MAS triggers the Network M2M Node to initiate termination is not covered in the present document. As far as the mId interface is concerned the Network M2M Node is the entity that appears to initiate the termination from network side.

The entity initiating the erase procedure (D/G M2M Node or Network M2M Node) shall transmit a PANA-Termination-Request (PTR) to the other end. This message shall include following AVPs: M2M-Usage-Type indicating "M2M Bootstrap-Erase", and M2M-Erase-Token. The M2M-Erase-Token is used for proving and verifying the authority of the sender. The M2M-Erase-Token can only be generated by an entity that knows the Kmr (i.e. the D/G M2M Node, the MSBF, or the MAS). If the PTR is accepted, then the receiver shall transmit a PANA-Termination-Answer (PTA) to the initiator of the procedure. Similar to the PTR, the PTA shall include following AVPs: M2M-Usage-Type indicating "M2M Bootstrap-Erase", and M2M-Erase-Token.

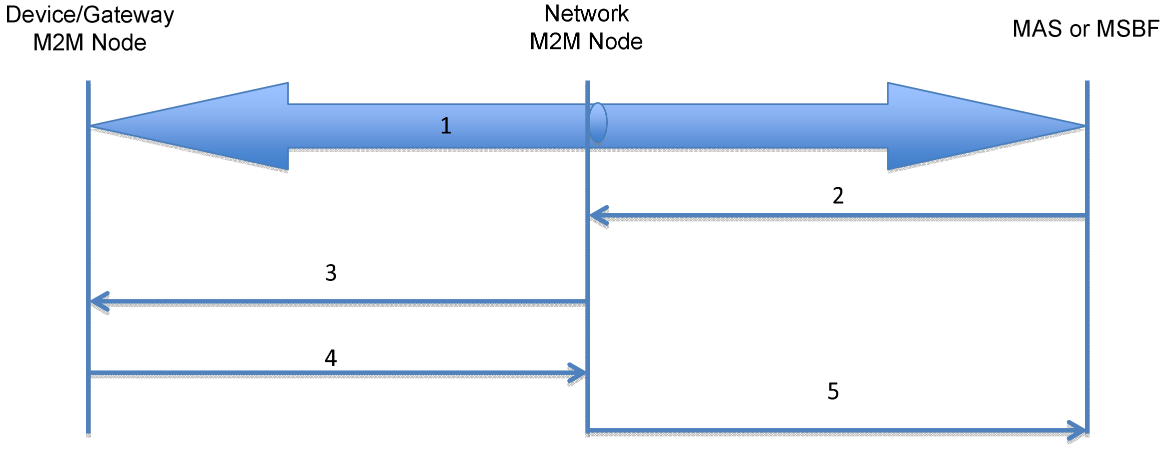


Figure 6.4: MAS or MSBF-initiated M2M Bootstrap-Erase Procedure

Figure 6.4 depicts the call flow for MAS or MSBF-initiated M2M Erase Procedure.

Step 1. M2M Service Bootstrap/Connect Procedure.

For the MAS-initiated case, this step is the M2M Service Connect Procedure. The D/G M2M Node and the MAS shall have already executed the M2M Service Connect Procedure and there shall be a PANA session between the D/G M2M Node and the Network M2M Node over which PTR/PTA messages can be exchanged (see steps 3 and 4).

For the MSBF-initiated case, this step is the M2M Service Bootstrap Procedure. The D/G M2M Node and the MSBF shall have already executed the M2M Service Bootstrap Procedure and there shall be a PANA session between the D/G M2M Node and the Network M2M Node over which PTR/PTA messages can be exchanged (see steps 3 and 4).

Step 2. Erase Request sent to Network M2M Node.

For the MAS/MSBF to initiate the Bootstrap-Erase Procedure, it shall send an Erase Request to the Network M2M Node. Decision to initiate the Bootstrap-Erase Procedure belongs to the MAS/MSBF and the details are not included in the present document.

The Erase Request shall contain a payload called M2M-Erase-Token AVP. This token conveys the intent of the sender along with the cryptographic proof about the authentication and integrity of the token.

The M2M-Erase-Token shall contain the following information elements.

M2M Node ID: This information element shall contain the identifier of the D/G M2M Node that will be erased.

Key Index: This information element shall contain the Kmr index (I_{Kmr}).

Nonce: This information element shall contain a nonce value generated by the sender. The sender shall make sure the same nonce value is not used with the same secret key value before, either by itself or by the other end-point. The value may be generated randomly, sequentially, or by any other pattern.

Type: This information element shall contain a value that indicates the type of this request token. It shall be set to 1 ("Network-initiated Erase Request").

Hash: This information element shall contain a cryptographically generated hash value in order to provide origin authentication, integrity and replay protection for the M2M-Erase-Token.

The following formula shall be used for computing the hash value.

- Hash = HMAC-SHA256(Kmr, M2M Node ID | Key Index | Nonce | Type)

Step 3. Erase Request sent to D/G M2M Node.

When the Network M2M Node receives the Erase Request, the Network M2M Node shall generate a PTR message that carries an M2M-Usage-Type AVP indicating "M2M Bootstrap-Erase", and an M2M-Erase-Token AVP carrying the received M2M-Erase-Token. The Network M2M Node shall send the PTR to the D/G M2M Node.

Step 4. Erase Request processed and Erase Response sent to Network M2M Node.

When the D/G M2M Node receives the M2M-Erase-Token AVP, then the D/G M2M Node shall verify the hash before accepting the request as authentic.

D/G M2M Node shall check the Nonce value in the M2M-Erase-Token AVP to make sure the Nonce value was not used previously for the same M2M Node ID and Key Index (I_{Kmr}). If the Nonce value was used before, then the D/G M2M Node shall ignore the request token.

The D/G M2M Node shall retrieve the Kmr by using the M2M Node ID and Key index (I_{Kmr}) for a look up from the D/G M2M Node's local key repository. If there is no matching Kmr, then the D/G M2M Node shall ignore the request token.

If a matching Kmr key is found, then this Kmr shall be used with the same formula as above (in Step 2) in order to generate a Hash value. If the computed Hash does not match the Hash value in the request, then the D/G M2M Node shall ignore the request token.

If the Hash is a match, then the D/G M2M Node shall accept this incoming request as a valid one. In this case, the D/G M2M Node shall make a decision on how to process the request based on the D/G M2M Node's local policy. The method for making the decision and configuring the local policy are not included in the present document. For example, a D/G M2M Node can configured to reject such requests. If the D/G M2M Node accepts executing the Bootstrap-Erase Procedure, then the D/G M2M Node shall delete its bootstrapped state after sending the response back. The D/G M2M Node may put a delay before deleting such state in case the response is not received by the MAS/MSBF and the MAS/MSBF retransmits the request (whose processing requires the state be present).

D/G M2M Node shall send a PANA-Termination-Answer (PTA) to the Network M2M Node that carries M2M-Usage-Type AVP indicating "M2M Bootstrap-Erase", and M2M-Erase-Token AVP.

M2M-Erase-Token shall contain the following information elements.

M2M Node ID: This information element shall contain the identifier of the D/G M2M Node being erased.

Key Index: This information element shall contain the Kmr index (I_{Kmr}).

Nonce: This information element shall contain the Nonce value copied from the received request.

Type: This information element shall contain a value that indicates the type of response. It may indicate values corresponding to "Erase Successful", "Erase rejected due to local policy", etc.

Hash: This information element shall contain a cryptographically generated hash value in order to provide origin authentication, integrity and replay protection for the M2M-Erase-Token.

The following formula shall be used for computing the hash value.

- Hash = HMAC-SHA256(Kmr, M2M Node ID | Key Index | Nonce | Type)

Step 5. Erase Response sent to MAS/MSBF.

The Network M2M Node shall relay the M2M-Erase-Token to the MAS/MSBF.

When the MAS/MSBF receives the response, the MAS/MSBF shall verify the hash before accepting the response as authentic.

The MAS/MSBF shall compare the Nonce received in the response with the Nonce transmitted in the request. If the Nonce values are not the same, then the MAS/MSBF shall ignore the response token.

The MAS/MSBF shall retrieve the Kmr by using the M2M Node ID and Key index (I_{Kmr}) for a look up from its local key repository. If there is no matching Kmr, then the MAS/MSBF shall ignore the response token.

If a matching key is found, it shall be used with the same formula as in Step 4 in order to generate a Hash value. If the computed Hash does not match the received Hash, then the MAS/MSBF shall ignore the response token.

If the Hash is a match, then the MAS/MSBF shall accept this incoming response as a valid one. In this case the MAS/MSBF shall decide what to do with the bootstrapped state associated with the D/G M2M Node based on the received Type and the MAS/MSBF's policy. The method for making decision is outside not included in the present document.

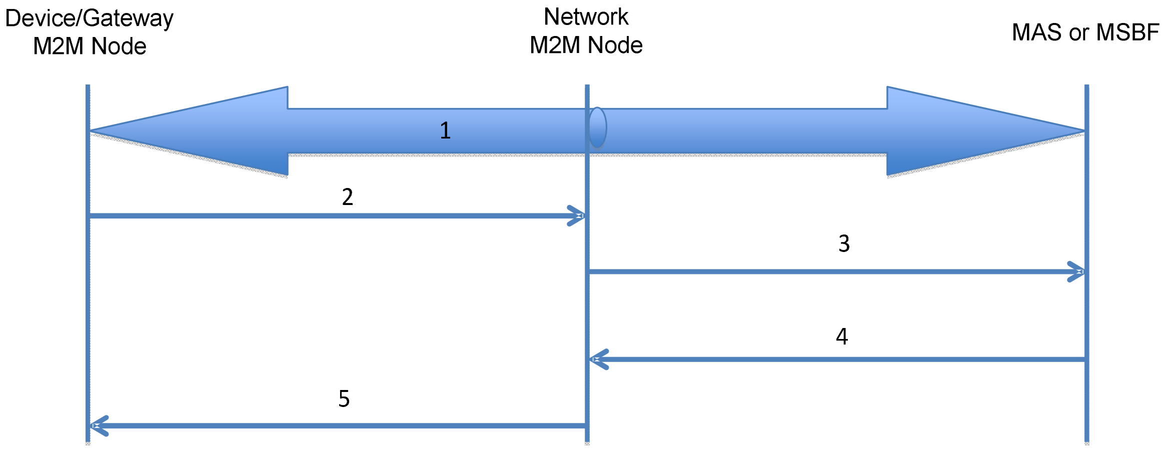


Figure 6.5: D/G M2M Node-initiated M2M Bootstrap-Erase Procedure

Figure 6.5 depicts the call flow for D/G M2M Node-initiated M2M Bootstrap-Erase Procedure.

Step 1. M2M Service Bootstrap/Connect Procedure.

When the Bootstrap-Erase Procedure is initiated by the D/G M2M Node, the initiator shall decide the target of this procedure: Either the MSBF or the MAS. How that target is decided is not included in the present document.

If the target is the MSBF, then the Step 1 corresponds to M2M Service Bootstrap Procedure. The PTR/PTA messages (see steps 2 and 5) shall be sent over the PANA session that is established for the M2M Service Bootstrap Procedure.

If the target is the MAS, then the Step 1 corresponds to M2M Service Connect Procedure. The PTR/PTA messages (see steps 2 and 5) shall be sent over the PANA session that is established for the M2M Service Connection Procedure.

Step 2. Erase Request sent to Network M2M Node.

For the D/G M2M Node to initiate the Bootstrap-Erase Procedure, the D/G M2M Node shall send a PANA-Termination-Request (PTR) message to the network. Decision to initiate the Bootstrap-Erase Procedure belongs to the D/G M2M Node and the details are not included in the present document.

The PTR message shall carry an M2M-Usage-Type AVP indicating "M2M Bootstrap-Erase", and an M2M-Erase-Token AVP.

The M2M-Erase-Token shall contain the following information elements.

M2M Node ID: This information element shall contain the identifier of the D/G M2M Node that will be erased.

Key Index: This information element shall contain the Kmr index (I_{Kmr}).

Nonce: This information element shall contain a nonce value generated by the sender. The sender shall make sure the same nonce value is not used with the same secret key value before, either by itself or by the other end-point. The value may be generated randomly, sequentially, or by any other pattern.

Type: This information element shall contain a value that indicates the type of this request token. It shall be set to 2 ("D/G-initiated Erase Request").

Hash: This information element shall contain a cryptographically generated hash value in order to provide origin authentication, integrity and replay protection for the M2M-Erase-Token.

The following formula shall be used for computing the hash value.

- Hash = HMAC-SHA256(Kmr, M2M Node ID | Key Index | Nonce | Type)

Step 3. Erase Request sent to MAS/MSBF.

When the Network M2M Node receives the Erase Request, it shall extract the M2M-Erase-Token from the M2M-Erase-Token AVP and relay the M2M-Erase-Token to the MAS/MSBF.

Step 4. Erase Request processed and Erase Response sent to Network M2M Node.

When the MAS or MSBF receives the M2M-Erase-Token AVP, it shall verify the hash before accepting the request as authentic.

The MAS/MSBF shall check the Nonce value in the received M2M-Erase-Token to make sure the Nonce value was not used previously for the same M2M Node ID and Key Index (I_{Kmr}). If the Nonce value was used before, then the MAS/MSBF shall ignore the request token.

The MAS/MSBF shall retrieve the Kmr by using the M2M Node ID and Key index (I_{Kmr}) for a look up from the MAS/MSBF's local key repository. If there is no matching Kmr, then the MAS/MSBF shall ignore the request token.

If a matching Kmr key is found, then this Kmr shall be used with the same formula as above in order to generate a Hash value. If the computed Hash does not match the Hash value in the request, then the MAS/MSBF shall ignore the request token.

If the Hash is a match, then the MAS/MSBF shall accept this incoming request as a valid one. In this case, the MAS/MSBF shall make a decision on how to process the request based on the MAS/MSBF's policy. For example, a MAS/MSBF can configured to reject such requests. The method for making the decision and configuring the policy are outside the scope of the present document. If the MAS/MSBF accepts executing the Bootstrap-Erase Procedure, then the MAS/MSBF shall delete the bootstrapped state after sending the response back. The MAS/MSBF may put a delay before deleting such state in case the response is not received by the D/G M2M Node and the D/G M2M Node retransmits the request (whose processing requires the state be present).

The MAS/MSBF shall send Erase Response to the Network M2M Node. This message shall contain M2M-Erase-Token AVP. M2M-Erase-Token shall contain the following information elements.

M2M Node ID: This information element shall contain the identifier of the D/G M2M Node being erased.

Key Index: This information element shall contain the Kmr index (I_{Kmr}).

Nonce: This information element shall contain the Nonce copied from the received request.

Type: This information element shall contain a value that indicates the type of response. It may indicate values corresponding to "Erase Successful", "Erase rejected due to local policy", etc.

Hash: This information element shall contain a cryptographically generated hash value in order to provide origin authentication, integrity and replay protection for the M2M-Erase-Token.

The following formula shall be used for computing the hash value.

- Hash = HMAC-SHA256(Kmr, M2M Node ID | Key Index | Nonce | Type)

Step 5. Erase Response sent to D/G M2M Node.

The Network M2M Node shall relay the M2M-Erase-Token to the D/G M2M Node over a PTA message that carries M2M-Usage-Type AVP indicating "M2M Bootstrap-Erase", and M2M-Erase-Token AVP.

When the D/G M2M Node receives the response, it shall verify the hash before accepting the response as authentic.

The D/G M2M Node shall compare the Nonce received in the response with the Nonce transmitted in the request. If the Nonce values are not the same, then the D/G M2M Node shall ignore the response token.

The D/G M2M Node shall retrieve the Kmr by using the M2M Node ID and Key index (I_{Kmr}) for a look up from its local key repository. If there is no matching Kmr, then the D/G M2M Node shall ignore the response token.

If a matching key is found, it shall be used with the same formula as in Step 4 in order to generate a Hash value. If the computed Hash does not match the received Hash, then the D/G M2M Node shall ignore the response token.

If the Hash is a match, then the D/G M2M Node shall accept this incoming response as a valid one. In this case, the D/G M2M Node shall decide what to do with the bootstrapped state associated with the D/G M2M Node based on the received Type and the D/G M2M Node's policy. The method for making the decision and configuring the policy are not included in the present document.

6.3.1.2 EAP/PANA - IBAKE bootstrapping operations

If Identity Based Authenticated Key Exchange (IBAKE) as per EAP-IBAKE [48] and RFC 6267 [42] is used for performing service bootstrapping independently of access network credentials for D/G M2M Node, then the procedures defined in this clause shall be used.

The IBAKE bootstrapping procedure shall be executed between D/G M2M Node and MSBF over the mId interface under direct management of the Network M2M Node. The selected Network M2M Node shall be authorized by the MAS for transporting the bootstrap transactions. In order to authorize transport of bootstrap transactions through the selected Network M2M Node, the D/G M2M Node shall be temporarily connected according to clause 7.3, using the pre-provisioned D/G M2M-Node-ID and shared secret as per TS 102 690 [2].

If the format and length of the pre-provisioned shared secret is not the same as the format and length requirement for Kmr (e.g. in use-cases where the shared secret is provided to the user in the form of a human-readable password), then the MAS shall use the same conversion function to the required format and length as was used by the D/G manufacturer while provisioning the password. This conversion function is outside the scope of the present document.

In order to execute secure IBAKE bootstrap procedure, certain IBE-specific parameters shall be present in the D/G M2M Node and the MSBF. Some of these parameters shall be pre-provisioned by the manufacturer, while others may be downloaded from the Network M2M Node into the D/G M2M Node. Successful completion of temporary M2M Service Connection produces security association that is used to protect download of IBE parameters required for execution of IBAKE bootstrapping. The list of required parameters and their secure download is defined in clause 6.3.1.2.1.3.

IBAKE bootstrap shall be executed using EAP-IBAKE [48] protocol transported over the mId by using PANA transport (RFC 5191 [33]) via the Network M2M Node as defined in clause 6.3.1.2.2. As the result of EAP-IBAKE execution, the key material shall be provided by the EAP for generating the M2M Root key Kmr for the M2M service layer.

The generated key Kmr shall then be provided by the MSBF to the MAS, as defined in clause 6.3.1.1.1.

6.3.1.2.1 Provisioning of IBE specific parameters

In order for D/G M2M Node and MSBF to execute EAP-IBAKE [48], they shall be provisioned with two sets of parameters. The first set parameters shall be provisioned at the manufacture, and it consists of publicly known mutually agreed-to parameters. The second set includes individual private parameters known only to their holder. These parameters may be pre-provisioned at manufacture, or may be securely downloaded by using procedures in this clause. Private parameters shall be provisioned securely and kept in a Secured Environment.

6.3.1.2.1.1 Provisioning of IBE specific Parameters into MSBF

Provisioning of the IBE parameters defined in clause 12.1.10 into the MSBF is out of scope of the present document.

6.3.1.2.1.2 Provisioning of IBE specific Parameters into D/G M2M Node

Provisioning of the IBE parameters defined in clause 12.1.10 into the D/G M2M Node by the manufacturer is out of scope of the present document.

When the IBE parameters defined in clause 12.1.10 are provisioned into the D/G M2M Node using secure PANA download upon successful mutual authentication, then procedures defined in clause 6.3.1.2.1.3 shall be used.

6.3.1.2.1.3 Online secure provisioning of IBE material to D/G M2M Node using PANA

The temporary M2M Service Connection procedure not only authenticates the end-points but also delivers the IBE parameters from the Network M2M Node to the D/G M2M Node. These parameters are needed for the subsequent EAP-IBAKE execution.

The final PAN message that carries the EAP-Success shall carry the M2M-IBE-Params AVP encapsulated inside an M2M-Encr-Encap AVP. The following key shall be used with the M2M-Encr-Encap AVP for the encryption:

- K_{PE} = Hash (MSK, "AVP Encryption Key").
- Hash is HMAC-SHA256.
- MSK is the Master Session Key as per RFC 3748 [22] generated by the EAP authentication method. MSK shall be made available to the D/G M2M Node by the EAP peer implementation and to the Network M2M Node by the EAP authenticator implementation.

I_{KPE} is the key index referring to K_{PE}. Its value shall be set to the value of I_{KR}. Lifetime of K_{PE} shall be set to the PANA Session-Lifetime assigned by the Network M2M Node.

6.3.1.2.1.4 Required IBE parameters used in IBAKE-based bootstrapping

The D/G M2M Node and MSBF shall be each pre-configured with the set of public parameters defined in table 6.1.

Table 6.1

Parameter	Description	Value
IBEEC	Elliptic curve used for IBE encryption and decryption	Weil pairing (BF) $y^2 = x^3 + 1 \bmod p$ See RFC 5091 [32], section 5
MEC	Elliptic curve used for point multiplication	K-163 Koblitz NIST curve (EC2NGF163Koblitz/ K-163 / sect163k1) [48]

All other required functions, algorithms and parameters may be pre-provisioned at the factory, or provisioned to D/G M2M Node using the last PANA message as specified in clause 6.3.1.2.1.3. If the operator chooses to provision the D/G M2M Node with new values for these parameters, then the last PANA message shall be used for transporting these parameters in a secure fashion. If the D/G M2M Node receives the new values for these parameters, the D/G M2M Node shall use the newly received values to complete the bootstrapping procedure.

All these parameters are provided in RFC 5091 [32] and RFC 5408 [37], along with suggested implementation-specific values.

6.3.1.2.2 Secure IBAKE protocol

The IBAKE protocol shall be executed between the D/G M2M Node and the MSBF via the authorized Network M2M Node. IBAKE messages shall be carried using EAP as the EAP-IBAKE method [48]. The EAP messages shall be transported over the mId interface using PANA [33]. The MSBF shall act as the EAP server, the Network M2M Node shall act as the EAP authenticator, and the D/G M2M Node shall act as the EAP peer.

6.3.1.2.2.1 Identity Based Encryption functions

EAP-IBAKE protocol details can be found in EAP-IBAKE [48]. Table 6.2 contains IBE function definitions used for generating IBE material as well as functions within the EAP-IBAKE protocol carried over PANA, which are specific to the case of bootstrapping for ETSI M2M. See EAP-IBAKE [48] for all remaining values related to EAP-IBAKE.

Table 6.2

Function	Description	Value/Name
H1	Function used for generating K_PUB (IBE public key point) from EAP identity [48]: MSBF side: K_PUBs = H1(IDs). D/G M2M Node side: K_PUBp = H1(IDp) where EAP identities shall be derived as follows: IDs = MSBF-ID|UTCDate|M2M-SP-ID. IDp = D/G M2M-Node-ID|UTCDate|M2M-SP-ID.	See RFC 5091 [32]
ENCR	IBE encryption function. Shall perform Weil-pairing based IBE encryption using K_PUB.	See RFC 5091 [32], section 5
DECR	IBE decryption function. Performs IBE decryption of an IBE-encrypted message.	See RFC 5091 [32], section 5
POINT2STR	Function that transforms a point on an elliptic curve to a string.	See RFC 5091 [32], section 4.3.2
STR2POINT	Function that transforms a string (e.g. key, identity, etc.) to a point on an elliptic curve.	See RFC 5091 [32], section 4.4.1
PRF	Pseudo-random function that generates a key, using as inputs a key and a string. It is used for generating EAP EMSK, MSK, as well as other keys.	HMAC SHA-256 shall be used as per EAP-IBAKE [48], SHA [43]

6.3.1.2.2.2 Protocol message parameters

Message formats for EAP-IBAKE can be found in EAP-IBAKE [48], for EAP-IBAKE header and all EAP-IBAKE payloads associated with the protocol. Table 6.3 specifies fixed message field values that shall be contained within the messages exchanged with the EAP-IBAKE protocol carried over PANA, which are specific to the case of automated bootstrapping for ETSI M2M. See EAP-IBAKE [48] for all remaining values.

Table 6.3

Field	Description	Value/Name
Type	Contained within the EAP-IBAKE header	"EAPIBAKE"
Reserved	Contained within IBAKE-ID/Request/Response payload	'0'
Proposal	Contained within IBAKE-ID/Request/Response payload	NULL
NumProposals	Number of proposal fields in the payload	'0'
IDType	Identity Type	5 (ID_FQDN)
Identity	Identity field in EAP-IBAKE payload	IBAKE-ID/Request: IDs IBAKE-ID/Response: IDp

EAP-IBAKE failure payload and values, as well as EAP fragmentation shall be supported as per EAP-IBAKE [48].

6.3.1.3 EAP-TLS over PANA

If TLS-based authentication is desired with EAP/PANA, then EAP-TLS as per RFC 5216 [34] shall be used. EAP-TLS shall be transported over EAP/PANA just like any other EAP method as described in clause 6.3.1. Furthermore, ETSI M2M TLS/certificate-specific details are covered in clause 6.3.3.

6.3.2 M2M Service Bootstrap Procedure using TLS over TCP

6.3.2.1 Recap of M2M Service Bootstrap Procedure using TLS over TCP

The architecture level details for M2M Service Bootstrap Procedure using TLS over TCP are specified in clause 8.3 of TS 102 690 [2], noting that clause 8.3.3.5 of TS 102 690 [2] provides the details in common with the M2M Service Bootstrap Procedure using EAP-TLS.

This clause provides an overview of the description in TS 102 690 [2] to explain how to apply the additional details in the the present document.

Pre-Provisioning: The D/G M2M Node and MSBF shall be provided with appropriate credentials to enable mutual authentication during an M2M Service Bootstrap procedure using TLS over TCP. Stage 3 details are specified in clause 6.3.2.2.

The M2M Service Bootstrap Procedure using TLS over TCP occurs in three main steps (see clause 8.3.3.4.1 of TS 102 690 [2]):

1) **Mutual Authentication:** The D/G M2M Node and MSBF shall be (freshly) mutually authenticated using a TLS Handshake and TLS shall export a KmrWrapKey. Further details are specified in clause 6.3.2.3.

2) **Parameter Delivery to the D/G M2M Node:** The D/G M2M Node shall be provisioned (by the MSBF) with parameters, including the value of Kmr encrypted under KmrWrapKe. Further details are specified in clause 6.3.2.4.

3) **Parameter Delivery to the MAS:** The MSBF shall provide parameters to the MAS. The stage 3 details for this step are not included in the present document.

Many of the details for this M2M Service Bootstrap procedure are shared with otherM2M Service Bootstrap procedure, with the result that most of the specification for this M2M Service Bootstrap procedure simply refers to the appropriate common specifications.

6.3.2.2 Pre-Provisioning for M2M Service Bootstrap Procedure using TLS over TCP

NOTE: Clause 8.3.3.5.2 of TS 102 690 [2] describes the credentials to be pre-provisioned to the D/G.

The M2M Device/Gateway certificates shall conform to the specification in clause 6.3.3.3.1.1.

The MSBF certificates shall conform to the specification in clause 6.3.3.3.2.1.

TLS support requirements are specified in clause 6.3.3.2.

6.3.2.3 Mutual Authentication for M2M Service Bootstrap Procedure using TLS over TCP

NOTE: Step 1 in clause 8.3.3.4.1 of TS 102 690 [2] specifies that the D/G M2M Node and MSBF authenticate each other using certificates as part of a TLS handshake over TCP.

The TLS handshake shall conform to the TLS handshake provided in clause 6.3.3.2.

MSBF processing of M2M Device/Gateway certificates shall conform to clause 6.3.3.3.1.2.

M2M Device/Gateway processing of MSBF certificates shall conform to clause 6.3.3.3.2.2.

Following successful TLS authentication, the MSBF and M2M Device/Gateway shall export KmrWrapKey by applying the TLS Exporter specification RFC 5705 [69] using the label "EXPORTER-ETSI-TC-M2M-Bootstrap" and length 32 with optional context value being omitted (i.e. no length and no value).

EXPORTER-ETSI-TC-M2M-Bootstrap has been registered with IANA for inclusion on the TLS parameters registry: http://www.iana.org/assignments/tls-parameters/tls-parameters.xml.

6.3.2.4 Parameter Delivery to D/G M2M Node for M2M Service Bootstrap Procedure using TLS over TCP

The D/G M2M Node and MSBF shall perform step 2 in clause 8.3.3.4.1 of TS 102 690 [2] by applying the M2M Service Bootstrap Parameter Delivery Procedure in clause 6.4, with specializations applicable to the M2M Service Bootstrapping procedure using TLS over TCP.

If the M2M Service Bootstrap Parameter Delivery Procedure succeeds, then the M2M Service Bootstrap Parameter Delivery Procedure outputs identical sets of Bootstrap parameters at both the D/G M2M Node and the MSBF.

If the M2M Service Bootstrap Parameter Delivery Procedure fails, then the M2M Service Bootstrap Parameter Delivery Procedure will output a description of the error that occurred. The MSBF and D/G M2M Node shall react to failure as follows:

1) **MSBF Error Actions:** If the M2M Service Bootstrap Parameter Delivery Procedure returns a failure for the MSBF, then the MSBF shall exit the M2M Service Bootstrap Procedure using TLS over TCP.

2) **D/G M2M Node Error Actions:** If the M2M Service Bootstrap Parameter Delivery Procedure returns a failure for the D/G M2M Node, then:

   a) The D/G M2M Node shall end the TLS Session (if the TLS session has not already been terminated).

   b) The D/G M2M Node shall exit the M2M Service Bootstrap Procedure using TLS over TCP.

6.3.3 Specifications for TLS/Certificate-Based M2M Service Bootstrap Procedures

6.3.3.1 Introduction

The architecture for TLS/Certificate-Based M2M Service Bootstrap procedures is described in clause 8.3.3.5 of TS 102 690 [2]. These TLS/Certificate-Based M2M Service Bootstrap Procedures are the procedure using EAP-TLS over PANA (clause 6.3.1.3) and the procedure using TLS over TCP (clause 6.3.2).

Clause 6.3.3 of the present document specifies the details for M2M Device/Gateways, M2M Nodes in those M2M Device/Gateways and MSBF that support one or both of these M2M Service Bootstrap Procedures. The following details are provided herein:

- TLS Details.
- Certificate Considerations:
  - M2M Device/Gateway Certificate Considerations:
    - M2M Device/Gateway Certificate Specifications.

  - ▪ MSBF Processing Requirements for M2M Device/Gateway Certificates.

- MSBF Certificate Considerations:

  - ▪ MSBF Certificate Specifications.

  - ▪ M2M Device/Gateway Processing Requirements for MSBF Certificates.

6.3.3.2 TLS Details for TLS/Certificate-Based M2M Service Bootstrap Procedures

The bulk encryption algorithm for TLS (in these M2M Service Bootstrap procedures) shall use 256-bit encryption keys. The following TLS Versions are supported for these M2M Service Bootstrap procedures:

- TLS v1.2 (RFC 5246 [35]) with the following TLS Cipher suites:

  - TLS_RSA_WITH_AES_128_CCM [74] with client-certificate based authentication.

- TLS v1.1 (RFC 4346 [28]) with the following TLS Cipher suite:

  - TLS_RSA_WITH_AES_128_CBC_SHA (RFC 4346 [28]) with client-certificate based authentication.

The certificates used in the TLS Handshake shall conform to clause 6.3.3.3.

For a D/G-M2M-Node to support a TLS/Certificate-Based M2M Service Procedure, the D/G-M2M-Node shall support at least one of these combinations of TLS version and TLS cipher suite.

For an MSBF to support a TLS/Certificate-Based M2M Service Bootstrap Procedure, the MSBF shall support all combinations of TLS version and TLS cipher suite supported by the M2M Devices/Gateways with which the MSBF wishes to perform M2M Service Bootstrapping using TLS over TCP.

6.3.3.3 Certificate Considerations

6.3.3.3.1 M2M Device/Gateway Certificate Considerations

6.3.3.3.1.1 M2M Device/Gateway Certificate Specifications

NOTE 1: The M2M Device/Gateway certificate is used by the MSBF to authenticate D/G-M2M-Nodes instantiated in the M2M Device/Gateway with which the certificate is associated.

The M2M Device/Gateway certificate shall be pre-provisioned by the M2M Device/Gateway Manufacturer during the manufacturing of the M2M Device/Gateway. The private key associated with the M2M Device/Gateway shall be stored in a Secured Environment Domain as per TR 102 725 [i.1] on the M2M Device/Gateway.

NOTE 2: This Secured Environment Domain is considered to be part of the D/G-M2M-Node performing the M2M Service Bootstrap procedure.

The M2M Device/Gateway certificate shall be compliant to RFC 5280 [36]. In addition, the M2M Device/Gateway certificate shall conform to the following specifications:

1) The signature algorithm used by the CA to sign the certificate shall be "sha256WithRSAEncryption", and the RSA public key used for signing shall be at least 2 048 bits and shall be based on PKCS #1 version 2.1 defined in RFC 3447 [20].

NOTE 3: The algorithm identifier for "sha256WithRSAEncryption" is defined in RFC 4055 [24].

2) The issuer name shall not be empty and shall identify the name of the issuer as defined in RFC 5280 [36], section 4.1.2.4.

3) The subject public key shall use algorithm "rsaEncryption" for RSASSA-PSS in RFC 4055 [24], and the RSA public key value shall be at least 2 048 bits.

4) The subjectAltName extension shall be present for M2M Device/Gateway certificate and shall contain a pre-provisioned identity.

5) The M2M Device/Gateway certificate shall contain validity time and Validity encoding shall be as specified in RFC 5280 [36]. The period of validity of the M2M Device/Gateway certificate shall be congruent with the expected lifetime of the certificates for such type of Device/Gateway.

NOTE 4: The period of validity of the M2M Device/Gateway certificate is assigned at the discretion of the M2M Device/Gateway Manufacturer. The M2M Device/Gateway Manufacturer has to take into account the expected productive lifetime of the M2M Device/Gateway when assigning the period of validity for the certificate.

6.3.3.3.1.2 MSBF Processing of M2M Device/Gateway Certificates

The MSBF processing requirements for M2M Device/Gateway certificates are defined below:

- The D/G-M2M-Node shall not send certificate paths containing more than four certificates.
- The MSBF shall be able to support M2M Device/Gateway certificate paths containing up to four certificates. The intermediate certificates are obtained from the TLS Handshake "Certificate" message and the TTP CA certificate is obtained from a MSBF local stored of trusted TTP CA certificates.
- The MSBF shall check the validity time of all certificates in the M2M Device/Gateway certificate path, and reject certificate(s) that are either not yet valid or that are expired. The MSBF should reject certificate(s) with a period of validity congruent with the expected lifetime of the certificates for such type of Device/Gateway.

6.3.3.3.2 MSBF Certificate Considerations

6.3.3.3.2.1 MSBF Certificate Specifications

NOTE 1: The MSBF certificate is used by the D/G-M2M-Node to authenticate the MSBF as part of the TLS handshake.

The MSBF certificate shall have a certificate chain back to a TTP Root CA certificate or TTP Intermediate CA certificate present in the Security Environment Domain of the D/G M2M Node that performs this procedure.

NOTE 2: The private key associated with the MSBF certificate is expected to be stored securely.

The MSBF certificate profile shall be compliant to RFC 5280 [36]. In addition, the MSBF certificate shall comply with the following specifications:

1) The signature algorithm used by the CA to sign the certificate shall be "sha256WithRSAEncryption", and the RSA public key used for signing shall be at least 2 048 bits and shall be based on PKCS #1 version 2.1 defined in RFC 3447 [20].

NOTE 3: The algorithm identifier for "sha256WithRSAEncryption" is defined in RFC 4055 [24].

2) The issuer name shall not be empty and shall identify the name of the issuer as defined in clause 4.1.2.4 of RFC 5280 [36].

3) The subject public key shall use algorithm "rsaEncryption" for RSAES-OAEP in RFC 4055 [24], and the RSA public key value shall be at least 2 048 bits.

4) The subject name may be empty in MSBF certificates and shall not be empty in CA certificates.

5) The subjectAltName extension shall be present if this is a MSBF certificate, and shall contain the FQDN of the MSBF.

6) The MSBF certificate shall contain validity time and Validity encoding shall be as specified in RFC 5280 [36].

6.3.3.3.2.2 D/G-M2M-Node Processing of MSBF Certificates

The D/G-M2M-Node processing requirements for MSBF certificates are defined below:

1) The processing of the MSBF certificate by D/G-M2M-Node shall be compliant to RFC 5280 [36].

2) The MSBF shall not send certificate paths containing more than four certificates.

3) The M2M Device/Gateway shall be able to support MSBF certificate paths containing up to four certificates.

NOTE 1: The intermediate (MSBF) CA certificates and the MSBF certificate are obtained from the TLS Handshake "Certificate" message sent from the MSBF:

- The TTP root CA certificate or TTP intermediate CA certificate shall be obtained from a local store of trusted TTP CA certificates in M2M Device/Gateway in which the D/G-M2M-Node is instantiated. The lifetime of the certificates in the local store shall be congruent with the expected lifetime of the certificates for such type of Device/Gateway.

NOTE 2: The period of validity of the root CA certificate or TTP intermediate CA certificate is assigned at the discretion of the M2M Device/Gateway Manufacturer. The M2M Device/Gateway Manufacturer has to take into account the expected productive lifetime of the M2M Device/Gateway when assigning the period of validity for the certificate.

4) The D/G-M2M-Node shall check the validity time of the MSBF certificates, and reject certificates that are either not yet valid or that are expired.

5) The D/G-M2M-Node shall perform MSBF Certificate Status Verification as per TS 102 690 [2]).

6.4 M2M Service Bootstrap Parameter Delivery Procedure For Procedures using HTTP

6.4.1 Overview

The M2M Service Bootstrap Parameter Delivery Procedures is used to securely deliver M2M Service Bootstrap Parameters from the MSBF to the D/G M2M Node, upon request from the D/G M2M Node, as part of the following M2M Service Bootstrap procedures:

- M2M Service Bootstrap procedure using GBA (clause 6.2.1).
- M2M Service Bootstrap procedure using TLS over TCP (clause 6.3.2).

The description to this procedure utilizes the mapping of primitives described in annex A. This procedure considers the M2M Service Bootstrap Parameters to be represented in a "virtual" resource called bootstrapParamSet which is identified by a default URI at the MSBF. The D/G M2M Node sends a request to execute the resource at this URI along with information that the MSBF requires to select the M2M Service Bootstrap Parameters. The MSBF processes this request and (as if the M2M Service Bootstrap Parameters really were present at this URI) the MSBF forms a representation of the M2M Service Bootstrap Parameters and returns this representation to the D/G M2M Node.

The following protocol mappings and resource representations shall be applied:

- The primitives shall be encoded using the protocol mapping annex for HTTP binding in annex C.
- The resource representation shall conform to the content types in annex B:
  - An MSBF that supports this procedure shall support all application/xml content types in annex B.
  - A D/G M2M Node that supports this procedure shall support at least one of the application/xml content types in annex B.

6.4.2 bootstrapParamSet Resource

NOTE: The only management procedure applicable to the bootstrapParamSet resource is the execute method, and consequently, the only message that can contain a bootstrapParamSet resource is a response primitive.

6.4.2.1 bootstrapParamSet Resource URI

The URI of the bootstrapParamSet resource shall be of the form:

- <MSBF-FQDN> "/bootstrapParamSet".

where:

- <MSBF-FQDN> This is the FQDN of the MSBF.

6.4.2.2 bootstrapParamSet Resource Attributes

The bootstrapParamSet resource shall contain the attributes in table 6.4. The format of these attributes shall conform to clauses 11.5 and 12.2.

Table 6.4: bootstrapParamSet Resource Attributes

AttributeName	Mandatory/Optional	Description
securityM2MNodeId	M	M2M-Node-ID assigned by the MSBF to the D/G M2M Node. See table 12.15.
securityKmrIndex	M	Kmr-Index for the Kmr. See table 12.15.
securityLifetime	M	Kmr lifetime, See table 12.15.
securityMasFqdn	M	FQDN of the MAS for which the Kmr key is destined. See table 12.15.
securityEncryptedM2MKey	O	Value of the Kmr, encrypted using AES-256 Key Wrap algorithm under the KmrWrapKey exported from the TLS master_secret. The attribute is present only if performing M2M Service Bootstrap procedure using TLS over TCP. See table 12.15.
sclId	O	SCL-ID assigned to the SCL in the D/G M2M Node as a result of this procedure. See table 11.36.
sclIdList	O	List of NSCL-ID's that the SCL in the D/G M2M Node may use for next point of contact. See table 12.15.

6.4.3 M2M Service Bootstrap Parameter Delivery Procedure Primitives

6.4.3.1 bootstrapParamSetExecuteRequestIndication

This request is used by the D/G M2M Node to request the MSBF to provide Service Bootstrap Parameters to the D/G M2M Node. The primitive shall comply with table 6.5. The format of the primitive attributes shall conform to clauses 11.6 and 12.3.

Table 6.5: bootstrapParamSetExecuteRequestIndication

bootstrapParamSetExecute Primitive: bootstrapParamSetExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
targetID	M	The URI of the bootstrapParamSet. See clause 6.4.2.1
primitiveType	M	BOOTSTRAP_PARAM_DELIVERY_EXECUTE
securityM2MNodeId	O	M2M-Node-ID of the D/G M2M Node, if already known to the D/G M2M Node
sclId	O	SCL-ID of the SCL in the D/G M2M Node, if already known to the D/G M2M Node
securityM2MSPId	O	The M2M Service Provider Identity value. This attribute shall be present if the D/G M2M Node intends to limit the procedure to the identified M2M Service Provider

6.4.3.2 bootstrapParamSetExecuteResponseConfirm (successful case)

This primitive confirms the establishment of an M2M Root Key (Kmr), and includes a resource that contains the parameters associated with the established M2M Service Bootstrap. The primitive shall comply with table 6.6. The format of the primitive attributes shall conform to clause 11.6.

Table 6.6: bootstrapParamSetExecuteResponseConfirm (successful case)

bootstrapParamSetExecute Primitive: bootstrapParamSetExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	BOOTSTRAP_PARAM_DELIVERY_EXECUTE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
bootstrapParamSet	M	Contains the M2M Service Bootstrap Parameters

6.4.3.3 bootstrapParamSetExecuteResponseConfirm (unsuccessful case)

The bootstrapParamSetExecuteRequestIndication primitive triggers this response. The primitive shall comply with table 6.7. The format of the primitive attributes shall conform to clause 11.6.

Table 6.7: bootstrapParamSetExecuteResponseConfirm (unsuccessful case)

bootstrapParamSetExecute Primitive: bootstrapParamSetExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	BOOTSTRAP_PARAM_DELIVERY_EXECUTE_RESPONSE
errorInfo	M	Provides error information

6.4.4 MSBF Filtering of Received bootstrapParamSetExecuteRequestIndication Primitives

The MSBF shall apply the following filtering to any received bootstrapParamSetExecuteRequestIndication primitives:

1) The MSBF shall process only bootstrapParamSetExecuteRequestIndication primitives that are either:

   a) (Applicable to M2M Service Bootstrap Procedure using TLS over TCP only) Received over a TLS Session over TCP established as specified in clause 6.3.2.3. The MSBF shall associate such bootstrapParamSetExecuteRequestIndication primitives with the pre-provisioned D/G identity in the certificate received from the D/G during the TLS handshake.

   b) (Applicable to M2M Service Bootstrap Procedure using GBA only) Received in a HTTP request that passes HTTP Digest Authentication as specified in clause 6.2.1.2. The MSBF shall associate such bootstrapParamSetExecuteRequestIndication primitives with the GBA B-TID sent by the D/G as part of the HTTP Digest Authentication.

2) The MSBF shall respond to a bootstrapParamSetExecuteRequestIndication that is received under other circumstances by performing the following steps:

   a) The MSBF shall send a bootstrapParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo including statusCode set to "STATUS_PERMISSION_DENIED".

   b) The MSBF shall exit the procedure, returning the statusCode.

6.4.5 M2M Service Bootstrap Parameter Delivery Procedure Sequence of Events

The D/G M2M Node shall perform the following actions in sequence:

1) Compose bootstrapParamSetExecuteRequestIndication primitive (as defined in table 6.5).

2) Encode the bootstrapParamSetExecuteRequestIndication according to the appropriate protocol mapping indicating the content type(s) supported by the D/G M2M Node for the M2M Service Bootstrap Parameter Delivery Procedure (see clause 6.4.1).

3) The D/G M2M Node shall send bootstrapParamSetExecuteRequestIndication.

    a) (Applicable to M2M Service Bootstrap Procedure using TLS over TCP only) If the TLS session is terminated before the D/G is able to deliver the bootstrapParamSetExecuteRequestIndication, then the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

    b) There are a variety of other scenarios the D/G M2M Node exits this procedure without being able to deliver the corresponding bootstrapParamSetExecuteRequestIndication. In these cases, the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to delivery failure.

4) The D/G M2M Node shall wait for the corresponding bootstrapParamSetExecuteResponseConfirm from the MSBF:

    a) The D/G M2M Node shall process only bootstrapParamSetExecuteResponseConfirm primitives that are either:

        i) (Applicable to M2M Service Bootstrap Procedure using TLS over TCP only) Received over a TLS Session over TCP established with the MSBF as specified in clause 6.3.2.3.

        ii) (Applicable to M2M Service Bootstrap Procedure using GBA only) received in a HTTP request that passes HTTP Digest Authentication as specified in clause 6.2.1.2.

    b) (Applicable to M2M Service Bootstrap Procedure using TLS over TCP only) If the TLS session is terminated before the corresponding bootstrapParamSetExecuteResponseConfirm answer is received, then the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

    c) There are a variety of other scenarios in which the D/G M2M Node exits this procedure without receiving the corresponding bootstrapParamSetExecuteResponseConfirm answer. In these cases, the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to lack of response.

Upon receiving a bootstrapParamSetExecuteRequestIndication associated with a pre-provisioned D/G identity or GBA B-TID (see clause 6.4.4), the MSBF performs the following actions in sequence:

5) **MSBF syntax check:** The MSBF shall verify message validity. Message validity is specified by the appropriate protocol mapping (see clause 6.4.1):

    - If the message is not valid, then the MSBF shall perform the following steps:

        ▪ The MSBF shall send a bootstrapParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo containing "STATUS_BAD_REQUEST".

        ▪ The MSBF shall exit the procedure, returning the statusCode.

6) **Authorization:** The MSBF shall determine the MAS that will receive the Kmr if the M2M Service Bootstrap procedure is successful:

- If the MSBF determines that there is no MAS that will receive the Kmr if the M2M Service Bootstrap procedure is successful, ,(for example, if the MSBF does not support the M2M Service Provider indicated by the received securityM2MSPId primitive attribute) then the MSBF shall perform the following steps:

    - The MSBF shall send a bootstrapParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo including statusCode set to "STATUS_PERMISSION_DENIED".

    - The MSBF shall exit the procedure, returning the statusCode.

- If the MSBF determines that there is an MAS that will receive the Kmr if the M2M Service Bootstrap procedure is successful, then the MSBF shall proceed to the next step.

7) **Create bootstrapParamSet resource:** The MSBF shall form the virtual bootstrapParamSet resource representation conforming to the provided definitions in clause 6.4.2.2 by applying the following steps. The MSBF shall form the resource representation according to one of the content types identified in the bootstrapParamSetExecuteRequestIndication as being supported by the D/G M2M Node (see step 2).

   a) securityM2MNodeId: The MSBF selects an Assigned M2M-Node-ID for the D/G M2M Node. The MSBF shall set the value of the securityM2MNodeId attribute to the value of the Assigned M2M-Node-ID of D/G M2M Node.

   b) securityKmrIndex: The MSBF shall assign a Kmr-Index that has not been used previously with the Assigned M2M-Node-ID. The MSBF shall set the value of the securityKmrIndex attribute to the value of the assigned Kmr-Index.

   c) securityLifetime: The MSBF shall assign Kmr-Lifetime according to an (out of scope) policy agreed to by the MSBF and MAS:

      i) In the case of M2M Service Bootstrap procedure using GBA, the Kmr-Lifetime shall not exceed the GBA Ks lifetime (see TS 133 220 [6]).

      ii) The MSBF shall set the value of the securityLifetime attribute to the value of the assigned Kmr-Index.

   d) securityMasFqdn: The MSBF shall set the value of the securityMasFqdn attribute to be the FQDN of the MAS that will receive the Kmr if the M2M Service Bootstrap procedure is successful.

   e) securityEncryptedM2MKey: This attribute shall be present only if performing an M2M Service Bootstrap Procedure using TLS over TCP. In this case, the MSBF shall select a random secret 256-bit Kmr and shall set the value of securityEncryptedM2MKey to the value of Kmr, encrypted using AES-256 Key Wrap algorithm [70] under the KmrWrapKey exported from the TLS master_secret:

      i) (Applicable to M2M Service Bootstrap Procedures using GBA only).

      In this case Kmr is derived from the GBA Ks_(int/ext)_NAF, as discussed in clause 6.2.1.3, and Kmr shall not be transmitted.

   f) (Optional) sclId: The MSBF may select an Assigned SCL-ID. If there is a SCL-ID received in the bootstrapParamSetExecuteRequestIndication and the MSBF selects an Assigned SCL-ID, there is no restriction that the Assigned SCL-ID is identical to the received SCL-ID. If the MSBF selected an Assigned SCL-ID, then the MSBF shall set the value of the sclId attribute to the value of the Assigned SCL-ID.

   g) (Optional) sclIdList: The MSBF may set the value of the sclIdList to a list of NSCL-ID that the SCL in the D/G M2M Node can use as the next point of contact.

8) **Create successful bootstrapParamSetExecuteResponseConfirm:** The MSBF shall create a bootstrapParamSetExecuteResponseConfirm (successful case) primitive with statusCode indicating "STATUS_OK". The response shall include the representation of the virtual bootstrapParamSet resource. The message shall be encoded using the appropriate protocol mapping (see clause 6.4.1).

9) Send bootstrapParamSetExecuteResponseConfirm:

   a) (Applicable to M2M Service Bootstrap Procedure using TLS over TCP only) If the TLS session is terminated before the bootstrapParamSetExecuteResponseConfirm (successful case) can be delivered, then the MSBF shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

   b) If the MSBF is unable to deliver the bootstrapParamSetExecuteResponseConfirm (successful case) then the MSBF shall exit the procedure, returning an indication that the procedure failed due to delivery failure.

   c) If delivery is successful, then the MSBF shall exit the procedure, returning the parameters associated with the associated M2M Service Bootstrap.

Upon receiving a bootstrapParamSetExecuteResponseConfirm, the D/G M2M Node shall perform the following actions in sequence:

10) **D/G M2M Node syntax check.** The D/G M2M Node shall verify message validity. Message validity is specified by the appropriate protocol mapping (see clause 6.4.1) and by validating the received resource representation against the provided resource definition in clause 6.4.2.2:

   - If the message is not valid, the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to syntax problems with the bootstrapParamSetExecuteResponseConfirm.

11) **Process bootstrapParamSetExecuteResponseConfirm:** The D/G M2M Node shall examine the bootstrapParamSetExecuteResponseConfirm primitive attributes:

   a) If the primitive attributes include the errorInfo attribute (that is, in the unsuccessful case), then the D/G M2M Node shall exit the procedure, returning the errorInfo attribute.

   b) If the primitive attributes include the statusCode "STATUS_OK" (that is, in the successful case), then the D/G M2M Node shall apply AES-256 Key Wrap decryption [70] using KmrWrapKey to extract Kmr from the securityEncryptedM2MKey:

      ▪ If the AES-256 Key Wrap decryption is not successful, then the D/G M2M Node shall exit the procedure, returning an indication that the procedure failed due to failed AES-256 Key Wrap decryption.

      ▪ If the AES-256 Key Wrap decryption is successful, then the D/G M2M Node shall exit the procedure returning the mIdSecurityFlag attribute, Kmr and the other M2M Service Bootstrap parameters in the bootstrapParamSetExecute.

7 M2M Service Connection Procedures

7.1 General principles

The use of an M2M Service Connection Procedure described in the present document is optional.

If an M2M Service Connection procedure is not used, then access network security clause 8.1 shall be used for mId Security.

Whether or not M2M Service Connection is required for the communication is pre-configured in the D/G M2M Node and Network M2M Node.

NOTE: Upon each failure of a M2M Service Connection Procedure, the D/G M2M Node is recommended to increase the time that it waits before attempting an M2M Service Connection Procedure again. This time can be reset to its initial value upon successful M2M Service Connection Procedure.

The M2M Service Connection Procedures described in clauses 7.2.1 (GBA-based) and 7.2.2 (EAP/PANA based) assume that the M2M Service Provider and the Access Network Provider are the same or that they have established a trusted relationship.

When this is not the case, any of the M2M Service Connection Procedures described in clauses 7.3 (EAP/PANA based) or 7.4 (TLS - PSK based) may be used. If an M2M Service Provider supports Integrity Validation, then the M2M core side shall comply with the specification of clause 7.5.

7.2 M2M Service Connection Procedures leveraging access network credentials

7.2.1 M2M Service Connection Procedure based on GBA

As described in clause 8.4.4 of TS 102 690 [2], the GBA-based M2M Service Connection procedure shall start with the D/G M2M Node and the BSF carrying out GBA bootstrapping over the Ub interface. There are two modes of GBA: ME-based (GBA_ME) and UICC-based (GBA_U). The latter requires that the UICC is GBA aware. The BSF shall decide which mode to run based on the UICC capability indicated in the GBA user security settings (GUSS). TS 124 109 [5] defines stage 3 for the HTTP Digest AKA based implementation of the Ub interface.

After a successful GBA bootstrapping, the D/G M2M Node and the BSF share a security association which consists of a bootstrapping transaction identifier (B-TID) and key material (GBA Ks). This security association with the BSF may be used by the D/G M2M Node to securely connect to one or more Network M2M Nodes and derive a unique Kmc for each of the connections.

7.2.1.1 TLS-PSK with GBA bootstrapped security association

The M2M Device/Gateway and the Network M2M Node may use the GBA security association to set up a TLS tunnel. clause 5.3.3 of TS 124 109 [5] defines the TLS-PSK based implementation of bootstrapped security association usage over the Ua interface.

When the D/G M2M Node initiates the TLS handshake, it shall indicate to the Network M2M Node that it supports PSK-based TLS by adding one or more PSK ciphersuites to the ClientHello message. This message shall also contain the hostname of the Network M2M Node in the server_name extension.

The Network M2M Node selects one of the PSK-based ciphersuites offered by the D/G M2M Node and sends it back in the ServerHello message. The ServerKeyExchange message contains a constant string "3GPP-bootstrapping" as the PSK-identity hint to indicate that Ks_NAF in the case of GBA_ME or Ks_ext_NAF in the case of GBA_U is used for authentication and key exchange. See clause 7.2.1.1.1 for the calculation of Ks_NAF/Ks_ext_NAF. The Network M2M Node shall finish the reply to the D/G M2M Node by sending a ServerHelloDone message.

The D/G M2M Node then sends a ClientKeyExchange message with PSK-identity containing a prefix "3GPP-bootstrapping", a separator character ";" and the B-TID. The D/G M2M Node shall conclude the TLS handshake by sending the ChangeCipherSpec and Finished messages to the Network M2M Node.

The Network M2M Node shall extract B-TID from the ClientKeyExchange message and use it to retrieve Ks_NAF or Ks_ext_NAF and associated key lifetime from the BSF. The retrieval shall be done over the Zn interface as specified in TS 129 109 [8]. As a result, the D/G M2M Node and the Network M2M Node/NAF share the NAF-specific key which is to be used as the M2M Connection Key (Kmc).

The Network M2M Node shall conclude the TLS handshake by sending the ChangeCipherSpec and Finished message to the D/G M2M Node.

Annex F of TS 124 109 [5] gives signalling flows for TLS-PSK with GBA bootstrapped security association.

The D/G M2M Node and the Network M2M Node shall support the TLS profile as specified in clause 5.4.1 of TS 133 222 [7].

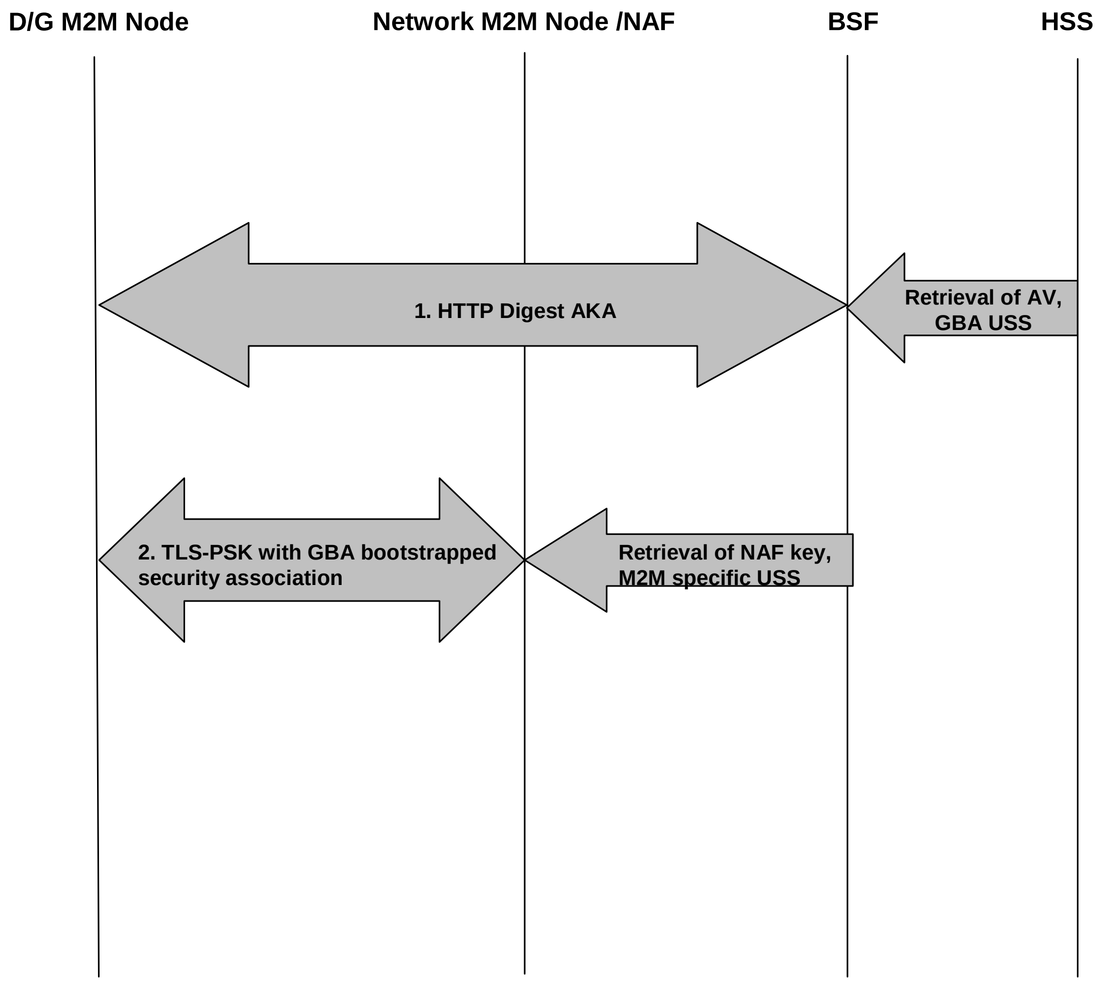

D/G M2M Node
Network M2M Node /NAF
BSF
HSS
1. HTTP Digest AKA
Retrieval of AV, GBA USS
2. TLS-PSK with GBA bootstrapped security association
Retrieval of NAF key, M2M specific USS

Figure 7.1: M2M Service Connection based on GBA

7.2.1.1.1 M2M Connection Key (Kmc) derivation

If GBA_ME is run, Ks_NAF as derived by the D/G M2M Node and the BSF shall be used as the M2M Connection Key (Kmc):

- Ks_NAF = KDF (Ks, "gba-me", RAND, IMPI, NAF_Id).

If GBA_U is run, two keys are derived:

- Ks_ext_NAF = KDF(Ks, "gba-me", RAND, IMPI, NAF_Id).
- Ks_int_NAF = KDF(Ks, "gba-u, RAND, IMPI, NAF_Id).

In this case Ks_ext_NAF shall be used as the Kmc.

In above KDF is the key derivation function as specified in annex B of TS 133 220 [6], and the key derivation parameters include the RAND (a part of the authentication vector), IMPI (for USIM, the IMPI is derived from IMSI), and the NAF_Id. The derived key is 256 bit long.

The NAF_Id is constructed as follows: NAF_Id = FQDN of the NAF | Ua security protocol identifier. As Network M2M Node is the NAF, the FQDN of the Network M2M Node shall be used. The Ua security protocol identifier is specified in annex H of TS 133 220 [6]. As PSK-based TLS according to TS 133 222 [7] is used as the Ua security protocol, the identifier shall be (0x01,0x00,0x01,yy,zz), where "yy,zz" is the protection mechanism CipherSuite code according to the defined values for PSK Ciphersuites for TLS RFC 4279 [27] and RFC 5487 [40].

7.2.2 M2M Service Connection Procedure Based On EAP/PANA with Access Network Credentials

This method applies to deployments where the M2M Service Provider and the Network Access Provider are the same or have a trusted relationship; so that EAP/PANA based connection procedures may use access network credentials.

Deployments that utilize SIM-based credentials with EAP-based M2M Connection Procedure shall use EAP-SIM [25] with EAP/PANA. Similarly, EAP-AKA [26] and EAP-AKA' [38] shall be used with EAP/PANA when AKA-based credentials need to be used with an EAP-based procedure. If the EAP-SIM or EAP-AKA or EAP-AKA' credentials are stored in a UICC [3] and shall not be exposed in the M2M device, then the UICC EAP framework specified in TS 102 310 [4] shall be used to avoid exposure of the UICC-based credentials used during the EAP authentication process.

EAP/PANA-based M2M Connection Procedure is agnostic to the authentication credentials and methods. Therefore, EAP-SIM and EAP-AKA or EAP-AKA' shall be carried over EAP/PANA according to the general procedure defined in clause 7.3.

7.3 M2M Service Connection Procedures using EAP/PANA

This clause describes the M2M Service Connection Setup and Tear-down Procedures using EAP/PANA.

7.3.1 M2M Service Connection Setup Procedure using EAP/PANA

EAP/PANA shall be used by the D/G M2M Node and the Network M2M Node for performing M2M Service Connection Setup procedure. This procedure connects the D/G M2M Node to a Network M2M Node by performing end-point authentication/authorization, parameter discovery, and key agreement.

EAP [22] shall be used for mutual authentication between the Device/Gateway and the MAS via the Network M2M Node. EAP-GPSK [39] shall be used as the EAP authentication method. PANA [33] shall be used as the transport protocol for EAP and M2M Service Connection parameters.

The D/G M2M Node shall implement the EAP peer functionality, the Network M2M Node shall implement the EAP authenticator functionality, and the MAS shall implement the EAP authentication server functionality [22] for executing EAP. Furthermore, the D/G M2M Node shall implement PANA Client (PaC) functionality, and the Network M2M Node shall implement PANA Authentication Agent (PAA) functionality [33] for transporting EAP and other parameters over PANA. There is no protocol mandate for the interface between the Network M2M Node and the MAS since that interface is not included in the present document. RADIUS [78] and Diameter [i.6] are examples of possible protocols that can be used over that interface.

The D/G M2M Node shall use the D/G M2M-Node-ID and the Kmr as its credentials with the EAP authentication method. These credentials may be pre-provisioned on the D/G M2M Node or dynamically bootstrapped (clause 6.3.1.1.1).

This procedure shall use the base PANA protocol [33] with additional AVPs. Only the new AVPs are described in the present document. Unless stated otherwise, base protocol behaviour and AVPs shall be used. For example, when the present document requires a PANA packet to include a newly-defined AVP, implementers shall take that AVP in addition to the standard AVPs as required by the base protocol [33] for the given packet type.

Figure 7.2 depicts the generic call flow for this procedure.

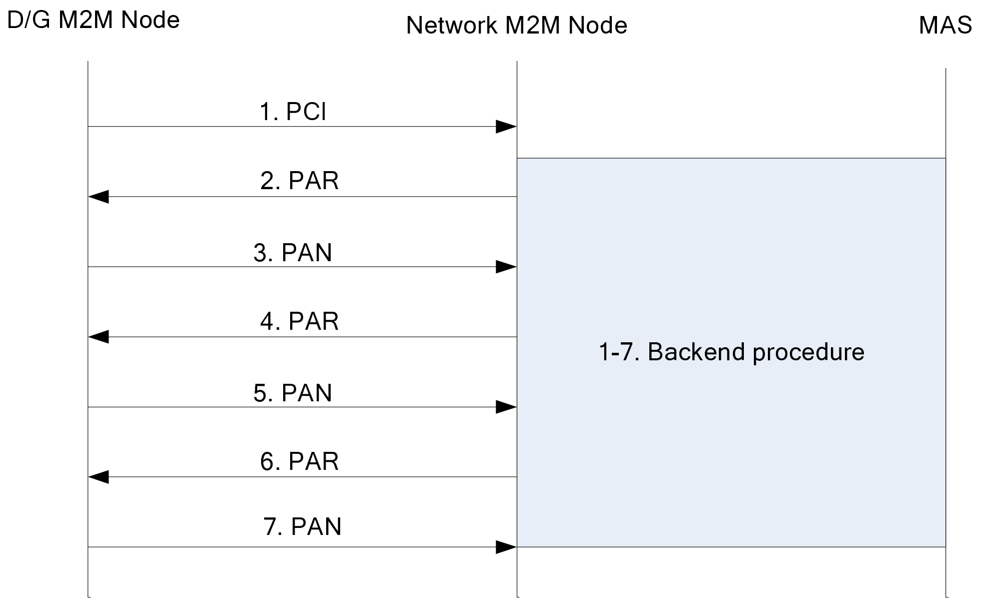


Figure 7.2: M2M Service Connection Setup Procedure using EAP/PANA

Steps 1-7 between the Network M2M Node and the MAS is not included in the present document. Therefore these steps are shown as block diagrams. Furthermore, procedures internal to the Network M2M Node are not shown.

Step 1. D/G M2M Node initiating PANA.

The D/G M2M Node shall initiate the M2M Service Connection Setup Procedure by sending a PANA-Client-Initiation (PCI) packet to the Network M2M Node. PCI shall include the following AVPs: M2M-Usage-Type carrying "M2M Connection Setup", and M2M-Node-ID carrying the D/G M2M Node identifier. PCI may also include the following AVP: M2M-SP-ID carrying the identifier of the M2M Service Provider. This optional AVP shall be present when the D/G M2M Node is limiting the procedure to a specific M2M Service Provider. For example, a D/G M2M Node that registers with the Network M2M Node for M2M-SP-ID = "M2M-Service-Provider.com". If this AVP is not present then this shall indicate that the D/G M2M Node is willing to connect to any M2M Service Provider's Network M2M Node.

The Network M2M Node shall ignore the incoming PCI if it contains a M2M-SP-ID value and the Network M2M Node is not configured to serve that domain.

Starting with Step 2, the Network M2M Node and MAS start signalling each other (for authentication and authorization) in coordination with the EAP/PANA communication between D/G M2M Node and Network M2M Node. No details are provided about that part since the interface between Network M2M Node and MAS is not included in the present document.

Step 2. First PANA packet from Network M2M Node.

The Network M2M Node sends the first PANA packet, PANA-Authentication-Request (PAR).

This PAR shall include the following AVPs: M2M-Usage-Type AVP indicating "M2M Connection Setup", M2M-SP-ID AVP. If the Network M2M Node wants to negotiate the mId security methods,, then M2M-MID-SEC AVP shall be included as well. If the Network M2M Node knows the parameters by some out-of scope mechanism (e.g. a pre-configured setup), then M2M-MID-SEC AVP may be omitted.

The value carried in M2M-SP-ID AVP shall be the same value as in the corresponding AVP carried in the PCI, if present. If the AVP is not present in the PCI, then the Network M2M Node shall set the value (carried in M2M-SP-ID AVP) to the identity of the M2M-SP operating the Network M2M Node.

The D/G M2M Node may ignore an incoming PAR if it carries a value in M2M-SP-ID AVP that does not match with the value sent in the PCI.

The first EAP AVP may be included in this packet or the subsequent packet from Network M2M Node [33].

Step 3. D/G M2M Node responds to Network M2M Node.

The Device/Gateway shall respond to the incoming PAR with a PANA-Authentication-Answer (PAN) packet, if PAR is not ignored at Step 2. This PAN shall include the following AVP: M2M-Usage-Type indicating "M2M Connection Setup", M2M-MID-SEC (only if it was also included in the PAR at step 2), M2M-KMR-Index carrying the key index of Kmr (I_{Kmr}) that will be used for authentication (omitted if relying on access network security, clause 8.1).

If the D/G M2M Node has received M2M-MID-SEC AVP at step 2, then it shall compare the received M2M-MID-SEC AVP value against the methods it supports and shall decide the method to be used (if any). The decision is based on a local policy and its details are outside the scope of the present document. The response from the D/G M2M Node shall indicate one method, or no method at all in case there is no match between what the Network M2M Node offers and what the D/G M2M Node supports. If no method is indicated by the D/G M2M Node, then the Network M2M Node shall interpret this as failure and not proceed with the subsequent steps. If the D/G M2M Node has not received M2M-MID-SEC AVP at step 2, then that means the D/G M2M Node and the Network M2M Node are using parameters that they have agreed to by some out-of scope mechanism.

Steps 4 and 5. Authentication.

If the M2M deployment is relying on access network security, then these steps shall be omitted. In other words, no authenticating EAP methods are used in this PANA session. Otherwise, these steps shall execute the EAP-GPSK [39] authentication by carrying EAP AVPs, unless EAP-SIM or EAP-AKA or EAP-AKA' are used as per clause 7.2.2.

Step 6. Final PAR from Network M2M Node.

This final PAR (Completion bit set) sent from the Network M2M Node shall signal the result of the M2M Service Connection Setup Procedure.

If the procedure is successful, the PAR shall include the following AVP: M2M-Connection-Result indicating success, M2M-Connection-ID carrying the connection identifier assigned to the D/G M2M Node by the Network M2M Node, and if not already assigned M2M-DSCL-ID carrying the D/GSCL-ID assigned to the D/GSCL by the NSCL.

If Object Security was selected during Step 2 and 3 and the network intends to use XML/HTTP, then the PAR shall also include M2M-XML-ALGOS AVP indicating the XML-ENC and/or XML-SIG algorithms supported by the network.

If the M2M deployment is relying on access network security, then Kmc and I_{Kmc} shall not be generated. Otherwise, both of the D/G M2M Node and the Network M2M Node shall generate Kmc according to the following formula upon successful M2M Service Connection:

- Kmc = Hash(MSK, "ETSI M2M Service Key" | D/G M2M-Node-ID)

where:

- Hash is HMAC-SHA256.
- MSK is the Master Session Key as per RFC 3748 [22] generated by the EAP authentication method. MSK shall be made available to the D/G M2M Node by the EAP peer implementation, and to the MAS by the EAP authentication server implementation. The MAS shall share the MSK with the Network M2M Node via the Network M2M Node - MAS interface.
- D/G M2M-Node-ID is the identifier of the D/G M2M Node assigned by the Network M2M Node.

The M2M Service Connection Setup Procedure can be executed multiple times. Either a new PANA session shall be established each time, or the new procedure shall be executed within the already established PANA session (when available). In either case the newly-generated Kmc shall be distinguishable from the earlier instances. The following key index is used for this purpose:

- I_{Kmc} = Session-Identifier | Key-Id

where:

- Session-Identifier is the PANA session identifier assigned by the Network M2M Node.
- Key-Id is the PANA key identifier assigned by the Network M2M Node.

The lifetime of Kmc shall be set to the PANA Session-Lifetime assigned by the Network M2M Node. Both the lifetime and the index of Kmc (I_{Kmc}) shall be stored along with the key itself.

If the procedure is not successful (e.g. due to EAP authentication or authorization failure), then the PAR shall include the following AVP: M2M-Connection-Result indicating failure.

Step 7. Final PAN from D/G M2M Node.

The D/G M2M Node sends a PAN in response to the PAR.

If the final PAR included M2M-XML-ALGOS, then the PAN shall also include M2M-XML-ALGOS AVP indicating the algorithms chosen by the D/G M2M Node. D/G M2M Node shall select exactly one algorithm for each algorithm-type marked as supported by the network. If the D/G M2M Node does not support any of the offered algorithms for a given algorithm-type, then no algorithm is selected for that type.

7.3.2 M2M Service Connection Tear-down Procedure using EAP/PANA

The M2M Service Connection Tear-down Procedure may be used for tearing down an M2M Service Connection between the D/G M2M Node and the Network M2M Node.

PANA Termination Procedure [33] shall be used for this purpose. Either the D/G M2M Node or the Network M2M Node may initiate the procedure.

The entity that desires to initiate the tear-down procedure (D/G M2M Node or Network M2M Node) shall transmit a PANA-Termination-Request (PTR) to the other end. This message shall include following AVP: M2M-Usage-Type indicating "M2M Connection Tear-down".

The receiver of the PTR shall execute the tear-down procedure upon verifying the authenticity of the PANA message. Execution of the tear-down procedure shall result in: tearing down any existing SCL registrations, if any; and deleting the connection state (Kmc, M2M-Connection-ID, etc.). The receiver of the PTR shall transmit a PANA-Termination-Answer (PTA) to the sender of the PTR. This message shall include following AVP: M2M-Usage-Type indicating "M2M Connection Tear-down".

7.4 M2M Service Connection Procedure based on TLS-PSK

7.4.1 Introduction

Clause 7.4.2 of the present document specifies the details for D/G M2M Nodes and MAS that support The M2M Service Connection procedure based on TLS-PSK. The following details are provided herein:

- TLS details.
- Sequence of events.
- Parameter delivery to D/G M2M Node.
- M2M Service Connection Parameter Delivery Procedure for TLS-PSK-based procedures.

7.4.2 TLS Details for M2M Service Connection Procedure Based On TLS-PSK

The following TLS Versions are supported for M2M Service Connection using TLS-PSK:

- TLS v1.2 (RFC 5246 [35]) with the following TLS Cipher suites:
    - TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 (RFC 5489 [71]).
- TLS v1.1 (RFC 4346 [28]) with the following TLS Cipher suites:
    - TLS_ ECDHE_PSK_WITH_AES_128_CBC_SHA (RFC 5489 [71]).

For a D/G M2M Node to support M2M Service Connection using TLS-PSK, the D/G M2M Node shall support at least one of these combinations of TLS version and TLS ciphersuite.

For an MAS to support M2M Service Connection using TLS-PSK, the MAS shall support all combinations of TLS version and TLS cipher suite supported by the D/G M2M Nodes to which the MAS wishes to perform M2M Service Connection procedures.

7.4.3 Sequence of events for M2M Service Connection Procedure based on TLS-PSK

The M2M Service Connection Procedure based on TLS-PSK shall perform the following steps (see clause 8.4.3.2 of TS 102 690 [2]):

1) **Mutual Authentication:** The D/G M2M Node and MAS shall (freshly) mutually authenticate each other using a TLS-PSK (RFC 4279 [27]) handshake. The TLS handshake shall conform to the TLS handshake provided in clause 7.4.2. Following successful TLS authentication, the MAS and M2M Device/Gateway shall export KmcWrapKey by applying the TLS Exporter specification RFC 5705 [69] using the label "EXPORTER-ETSI-TC-M2M-Connection" and length 32 with optional context value being omitted (i.e. no length and no value).

   EXPORTER-ETSI-TC-M2M-Connection has been registered with IANA for inclusion on the TLS parameters registry: http://www.iana.org/assignments/tls-parameters/tls-parameters.xml.

2) **Parameter Delivery to the D/G M2M Node:** The D/G M2M Node shall be provisioned (by the MAS) with M2M Service Connection parameters, including the M2M-Connection-ID . Additional details are specified in clause 7.4.4.

3) **D/G M2M Node Indicating M2M-Connection-ID:** The D/G M2M Node shall initiate the mId Security Procedures for channel security or object security indicated by the MAS as being preferred by the Network M2M Node. The D/G M2M Node shall provide the M2M-Connection-ID to the Network M2M Node in a key identifier during the mId Security Procedure. Each mId Security Procedure (clause 8) specifies how the D/G M2M Node provides the M2M-Connection-ID to the Network M2M Node.

4) **Network M2M Node Requests Parameters:** The Network M2M Node shall request parameters from the MAS. The details are not included in the present document.

5) **Parameter Delivery to the Network M2M Node:** The MAS shall provide the parameters to the Network M2M Node. The details for how the MAS provides the M2M Service Connection parameters to the Network M2M Node are not included in the present document.

6) **Network M2M Node Responds to D/G M2M Node:** The nature of the response depends on context in which the D/G M2M Node provided the M2M. Each mId Security Procedure (clause 8) specifies how the Network M2M Node responds to the D/G M2M Node during initialization of that mId Security Procedure. In mId Security Procedures providing channel security or object security, the Network M2M Node and D/G M2M use keys derived from Kmc.

7.4.4 Parameter Delivery to D/G M2M Node for M2M Service Connection Procedure based on TLS-PSK

The D/G M2M Node and MAS shall perform step 2 in clause 8.3.3.4.1 of TS 102 690 [2] by applying the M2M Service Connection Parameter Delivery Procedure in clause 7.4.5.

If the M2M Service Connection Parameter Delivery Procedure succeeds, then the procedure for the selected M2M Service Connection API outputs identical sets of Connection parameters at both the D/G M2M Node and the MAS.

If the M2M Service Connection Parameter Delivery Procedure fails, then the M2M Service Connection Parameter Delivery Procedure will output a description of the error that occurred. The MSBF and D/G M2M Node shall react to failure as follows:

1) **MAS Error Actions:** If the M2M Service Connection Parameter Delivery Procedure returns a failure for the MAS, then the MAS shall exit the M2M Service Connection Procedure based on TLS-PSK.

2) **D/G M2M Node Error Actions:** If the M2M Service Connection Parameter Delivery Procedure returns a failure for the D/G M2M Node, then:

   - The D/G M2M Node shall end the TLS Session(if the TLS Session has not already been terminated).

- The D/G M2M Node shall exit the M2M Service Connection Procedure Based on TLS-PSK.

7.4.5 M2M Service Connection Parameter Delivery Procedure For TLS-PSK-Based Procedures

7.4.5.1 Overview

The M2M Service Connection Parameter Delivery Procedure is used to deliver M2M Service Connection Parameters (as part of an M2M Service Connection Procedure based on TLS-PSK) from the MAS to the D/G M2M Node, upon request from the D/G M2M Node.

The description to this procedure utilizes the mapping of primitives described in annex A. This procedure considers the M2M Service Connection Parameters to be represented in a "virtual" resource called connectionParamSet which is identified by a default URI at the MAS. The D/G M2M Node sends a request to execute the resource at this URI (the URI includes the D/G M2M Nodes' M2M-Node-ID) and along with information that the MAS requires to select the M2M Service Connection Parameters. The MAS processes this request and (as if the M2M Service Connection Parameters really were present at this URI) the MAS forms a representation of the M2M Service Connection Parameters and returns this representation to the D/G M2M Node.

Within the M2M Service Connection Procedure based on TLS-PSK, these primitives shall be exchanged over a TLS session between the D/G M2M Node and MAS. The following protocol mappings and resource representations shall be applied:

- The primitives shall be encoded using the protocol mapping annex for HTTP binding for M2M REST resources annex C.
- The resource representation shall conform to the content types in annex B:
    - An MAS that supports this procedure shall support all application/xml content types in annex B.
    - A D/G M2M Node that supports this procedure shall support at least one of the application/xml content types in annex B.

7.4.5.2 connectionParamSet Resource

The only management procedure applicable to the connectionParamSet resource is the execute method, and consequently, the only message that may contain a connectionParamSet resource is a response primitive.

7.4.5.2.1 connectionParamSet Resource URI

The URI of the connectionParamSet resource shall be of the form:

- <MAS-FQDN> "/connectionParamSet"

where:

<MAS-FQDN> This shall be the FQDN of the MAS that is either pre-configured on the D/G M2M Node or established during an M2M Service Bootstrap Procedure.

7.4.5.2.2 connectionParamSet Resource Attributes

The connectionParamSet resource shall contain the attributes in table 7.1. The format of these attributes shall conform to clauses 11.5 and 12.2.

Table 7.1: connectionParamSet Resource Attributes

AttributeName	Mandatory/Optional	Description
securityConnectionId	M	M2M-Connection-Id for the established M2M Service Connection. See table 12.15.
securityKmcIndex	M	Kmc-Index for the Kmc. See table 12.15.
securityLifetime	M	Kmc lifetime. See table 12.15.
securityEncryptedM2MKey	M	Value of the Kmc for the established M2M Service Connection, encrypted using AES-256 Key Wrap algorithm under KmcWrapKey exported from the TLS master_secret. See table 12.15.
sclId	M	SCL-ID assigned to the SCL in the D/G M2M Node as a result of this procedure. See table 11.36.
securitymIdFlags	M	Indicates the mId Security methods to be used by the D/G M2M Node for mId Security. See table 12.15.
securityXmlAlgorithmFlags	O	This attribute carries XML security algorithms selected by the network. This attribute shall be present only if (a) the connectionParamSetExecureRequestIndication includes an securityXmlAlgorithmFlags attribute and (b) the network indicates to use object security in the securitymIdFlags in the resource. See table 12.15.

7.4.5.3 M2M Service Connection Parameter Delivery Procedure Primitives

7.4.5.3.1 connectionParamSetExecuteRequestIndication

This request is used by the D/G M2M Node to request the MAS to establish an M2M Service Connection with the Network M2M Node containing an identified NSCL. The primitive shall comply with table 7.2. The format of the primitive attributes shall conform to clauses 11.6 and 12.3.

Table 7.2: connectionParamSetExecuteRequestIndication

connectionParamSetExecute Primitive: connectionParamSetExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
targetID	M	The resource URI of the connectionParamSet. See clause 7.4.5.2.1
primitiveType	M	CONNECTION_PARAM_DELIVERY_EXECUTE_REQUEST
sclId	M	URI of the NSCL in the Network M2M Node for which the D/G M2M Node wishes to establish connection parameters
securitymIdFlags	M	Indicates the mId Security methods supported by the D/G M2M Node
securityXmlAlgorithmFlags	O	This attribute carries XML security algorithms supported by the D/G M2M Node This attribute shall be present only if the D/G M2M Node supports XML object security (clause 8.3.2)

7.4.5.3.2 connectionParamSetExecuteResponseConfirm (successful case)

This primitive confirms the establishment of an M2M Service Connection with the Network M2M Node containing an identified NSCL, and contains the parameters associated with the established M2M Service Connection. The primitive shall comply with table 7.3. The format of the primitive attributes shall conform to clause 11.6.

Table 7.3: connectionParamSetExecuteResponseConfirm (successful case)

connectionParamSetExecute Primitive: connectionParamSetExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONNECTION_PARAM_DELIVERY_EXECUTE_RESPONSE
statusCode	M	STATUS_OK
Resource attribute	**Mandatory/Optional**	**Description**
connectionParamSet	M	Contains the M2M Service Connection Parameters

7.4.5.3.3 connectionParamSetExecuteResponseConfirm (unsuccessful case)

The connectionParamSetExecuteRequestIndication primitive triggers this response. The primitive shall comply with table 7.4. The format of the primitive attributes shall conform to clause 11.6.

Table 7.4: connectionParamSetExecuteResponseConfirm (unsuccessful case)

connectionParamSetExecute Primitive: connectionParamSetExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONNECTION_PARAM_DELIVERY_EXECUTE_RESPONSE
errorInfo	M	Provides error information

7.4.5.4 M2M Service Connection Parameter Delivery Procedure Pre-Conditions

The following pre-conditions are necessary for a successful connectionParamSetExecute exchange:

1) The MAS shall be able to determine the set of mId Security Procedures supported by the identified Network M2M Node, and shall be able to determine the order of preference for any subset of those mId Security Procedures. The mechanism for achieving this is outside the scope of the present document.

2) The MAS shall be able to determine the combination of XML security algorithms (clause 8.3.2) supported by the identified Network M2M Node (the Network M2M Node may support no combinations of XML security algorithms), and shall be able to determine the order of preference for any subset of those combinations of XML security algorithm. The mechanism for achieving this is outside the scope of the present document.

7.4.5.5 MAS Filtering of Received connectionParamSetExecuteRequestIndication Primitives

The MAS applies the following filtering to any received BootstrapParamSetExecuteRequestIndication primitives:

- The MAS shall process only connectionParamSetExecuteRequestIndication primitives received over a TLS-PSK session established as per (clause 7.4.3). The MAS shall associate such connectionParamSetExecuteRequestIndication primitive with the D/G M2M Node identified by the psk_identity received in the TLS handshake.

- If the MAS receives a connectionParamSetExecuteRequestIndication primitive in any other circumstances, then the MAS shall perform the following steps:

    - The MAS shall send a connectionParamSetExecuteResponseConfirm primitive (unsuccessful case) with errorInfo including statusCode set to "STATUS_PERMISSION_DENIED".

    - The MAS shall exit the procedure returning the statusCode.

7.4.5.6 M2M Service Connection Parameter Delivery Sequence of Events

The D/G M2M Node shall perform the following actions in sequence:

1) The D/G M2M Node shall compose connectionParamSetExecuteRequestIndication primitive (as defined in table 7.2).

2) The D/G M2M Node shall encode the connectionParamSetExecuteRequestIndication according to the appropriate protocol mapping indicating the content type(s) supported by the D/G M2M Node for the M2M Service Connection Parameter Delivery Procedure (see clause 7.4.5.1).

3) The D/G M2M Node shall send connectionParamSetExecuteRequestIndication:

    a) If the TLS session is terminated before the D/G is able to deliver the connectionParamSetExecuteRequestIndication, then the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

    b) There are a variety of other scenarios the D/G M2M Node exits this procedure without being able to deliver the corresponding connectionParamSetExecuteRequestIndication. In these cases, the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to delivery failure.

4) The D/G M2M Node shall wait for the corresponding connectionParamSetExecuteResponseConfirm from the MAS:

    a) The D/G M2M Node shall process only connectionParamSetExecuteResponseConfirm messages received over a TLS-PSK session established as per (clause 7.4.3).

    b) If the TLS session is terminated before the corresponding connectionParamSetExecuteResponseConfirm answer is not received, then the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

    c) There are a variety of other scenarios in which the D/G M2M Node exits this procedure without receiving the corresponding connectionParamSetExecuteResponseConfirm answer. In these cases, the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to lack of response.

Upon receiving a connectionParamSetExecuteRequestIndication associated with a D/G M2M Node ID (see clause 7.4.5.5), the MAS performs the following actions in sequence:

5) **MAS syntax check.** The MAS shall verify message validity. Message validity is specified by the appropriate protocol mapping (see clause 7.4.5.1). If the message is not valid, then the MAS shall perform the following steps:

    - The MAS shall send a connectionParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo containing "STATUS_BAD_REQUEST".

    - The MAS shall exit the procedure, returning the statusCode.

6) **Authorization:** The MAS shall determine if the D/G M2M Node is authorized to establish an M2M Service Connection with the indicated Network M2M Node (that is the Network M2M Node containing the NSCL identified by the sclId attribute in connectionParamSetExecuteRequestIndication):

    a) If the indicated Network M2M Node is not an Network M2M Node for which the MAS is authorized to establish an M2M Service Connection, then the MAS shall perform the following steps:

        i) The MAS shall send a connectionParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo including statusCode set to "STATUS_NOT_FOUND".

        ii) The MAS shall exit the procedure, returning the statusCode.

    b) If the indicated Network M2M Node is not an Network M2M Node for which the MAS is authorized to establish an M2M Service Connection then:

        i) The MAS shall determine if the D /G M2M Node is authorized to establish an M2M Service Connection with the indicated Network M2M Node:

            1) If the MAS determines that the D/G M2M Node is not authorized to establish an M2M Service Connection with the indicated Network M2M Node, then the MAS shall perform the following steps:

                a) The MAS shall send a connectionParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo including statusCode set to "STATUS_FORBIDDEN".

b) The MAS shall exit the procedure, returning the statusCode.

2) If the MAS determines that the D/G M2M Node is authorized to establish an M2M Service Connection with the indicated Network M2M Node, then the MAS shall proceed to the next step.

7) **Select mId security method:** The MAS shall apply the following steps:

a) The MAS shall (from the received securitymIdFlags primitive attribute) determine the set of mId methods that are supported by the D/G M2M Node.

b) If (a) object layer security is indicated by the received securitymIdFlags primitive attribute, (b) the MAS knows that the Network M2M Node supports object layer security, and (c) the securityXmlAlgorithmFlags attribute is present in the received connectionParamSetExecuteRequestIndication, then the MAS shall determine the combinations of XML security algorithms that are mutually supported by the D/G M2M Node (as indicated by the received securitymIdFlags primitive attribute) and the indicated Network M2M Node (see clause 7.4.5.4):

i) If there are no combinations of XML security algorithms that are mutually supported by the D/G M2M Node and Network M2M Node, then object security shall be considered to be not mutually supported.

ii) If there is at least one mutually supported combinations of XML security algorithms that are mutually supported by the D/G M2M Node and Network M2M Node, then object security shall be considered as mutually supported and the MAS shall select the mutually supported combination of XML security algorithms that has highest preference for the Network M2M Node (see clause 7.4.5.4).

c) The MAS shall determine the set of mId methods that are mutually supported by the D/G M2M Node and the indicated Network M2M Node (see clause 7.4.5.4):

i) If they are no mutually-supported mId security methods, then the MAS shall perform the following steps:

1) The MAS shall send a connectionParamSetExecuteResponseConfirm (unsuccessful case) with errorInfo containing statusCode "STATUS_NOT_FOUND".

2) The MAS shall exit the procedure, returning the statusCode.

ii) If there is at least one mutually-supported mId method then the MAS shall select the mutually-supported mId method that has highest preference for the indicated Network M2M Node (see clause 7.4.5.4).

8) **Create connectionParamSet resource:** The MAS shall form the virtual connectionParamSet resource representation conforming to the provided definition in clause 7.4.5.2.2 by applying the following steps. The MAS shall form the resource representation according to one of the content types identified in the connectionParamSetExecuteRequestIndication as being supported by the D/G M2M Node (see step 2):

a) securityConnectionId and kmcIndex:

i) The MAS shall decide whether to establish a new M2M Service Connection or whether to extend a current valid M2M Service Connection, based on the following criteria:

1) If there is no current valid M2M Service Connection between the D/G M2M Node and the identified Network M2M Node, then the MAS shall establish a new M2M Service Connection.

2) If there is a current valid M2M Service Connection between the D/G M2M Node and the identified Network M2M Node, then either the MAS may extend the current valid M2M Service Connection or the MAS may establish a new M2M Service Connection.

ii) The MAS shall assign the M2M-Connection-ID and Kmc-Index according to the decision in step 8), a), i):

1) If the decision is to establish a new M2M Service Connection, then the MAS shall assign:

a) M2M-Connection-ID to be new Connection-ID that is unique within the scope of the Network M2M Node.

b) Kmc-Index to any value.

2) If the decision is to extend the current valid M2M Service Connection, then the MAS shall assign:

a) M2M-Connection-ID to be the M2M-Connection-ID of the current valid M2M Service Connection.

b) Kmc-Index to be a value that has not previously been used with the current M2M-Connection-ID.

The MAS shall set the values of securityConnectionId and securityKmcIndex to be the assigned values of M2M-Conneciton-ID and Kmc-Index respectively.

b) **securityLifetime:** The MAS shall assign Kmc-Lifetime according to policy of the MAS. The MAS shall set the value of securityLifetime to be the assigned value of Kmc-Lifetime.

c) **securityEncryptedM2MKey:** The MAS shall assign a random secret 256-bit Kmc. The MAS shall set the value of securityEncryptedM2MKey to be the assigned value of Kmc encrypted under KmcWrapKey, according to the AES-256 Key Wrap algorithm [70].

d) **sclId:** The MAS shall determine if the MAS already associates the D/G M2M Node with an SCL-ID (see clause 7.4.5.5):

i) If the MAS already associates the D/G M2M Node with an SCL-ID, then the MAS shall assign this SCL-ID.

ii) If the MAS does not already associate the D/G M2M Node with an SCL-ID, then the MAS shall select a SCL-ID to be assigned to the SCL in the D/G M2M Node. The MAS and NSCL need to have an (not currently specified) agreement regarding rules for constructing the SCL-ID.

The MAS shall set the value of sclId to be the assigned value of SCL-ID.

e) **SecuritymIdFlags:** The MAS shall set the value of securitymIdFlags to indicate the mutually-supported mId method determined in Step 7 "Select mId security method".

f) (optional) **securityXmlAlgorithmsFlags** (Applicable only if the MAS indicated use of object security in securitymIdFlags, and if the MAS selected a mutually supported combination of XML security algorithms in Step 7 "Select mId security method"). The MAS shall set the value of securityXmlAlgorithmsFlags to indicate the selected combination of XML security algorithms.

9) **Create successful connectionParamSetExecuteResponseConfirm:** The MAS shall create a connectionParamSetExecuteResponseConfirm (successful case) primitive with a statusCode indicating "STATUS_OK". The response shall include the representation of the virtual connectionParamSet resource. The message shall be encoded using the appropriate protocol mapping (see clause 7.4.5.1).

10) **Send connectionParamSetExecuteResponseConfirm:** The MAS shall attempt to deliver the connectionParamSetExecuteResponseConfirm (successful case) to the D/G M2M Node:

a) If the TLS session is terminated before the corresponding connectionParamSetExecuteResponseConfirm (successful case) can be delivered, then the MAS shall exit the procedure returning an indication that the procedure failed due to TLS session termination.

b) If the MAS is unable to deliver the connectionParamSetExecuteResponseConfirm (successful case) then the MAS shall exit the procedure, with an indication that the procedure failed due to delivery failure.

c) If delivery is successful, then the MAS shall exit the procedure, returning the parameters associated with the associated M2M Service Connection.

Upon receiving a connectionParamSetExecuteResponseConfirm, the D/G M2M Node shall perform the following actions in sequence:

11) **D/G M2M Node syntax check:** The D/G M2M Node shall verify message validity. Message validity is specified by the appropriate protocol mapping (see clause 7.4.5.1).and by validating the received resource representation against the provided schema definitions in clause 7.4.5.2.2.

    - If the message is not valid, then the D/G M2M Node shall exit the procedure returning an indication that the procedure failed due to syntax problems with the connectionParamSetExecuteResponseConfirm.

12) The D/G M2M Node shall process the connectionParamSetExecuteResponseConfirm primitive attributes according to the following rules:

    - If the primitive attributes include an errorInfo attribute (that is, in the unsuccessful case), then the D/G M2M Node shall exit the procedure, returning the errorInfo attribute.

    - If the primitives attributes includes the statusCode "STATUS_OK" (that is, in the successful case), then the D/G M2M Node shall apply AES-256 Key Wrap decryption [70] (using the KmcWrapKey) to extract Kmc from the securityEncryptedM2MKey:

        - If the AES-256 Key Wrap decryption is not successful, then the D/G M2M Node shall exit the procedure, returning an indication that the procedure failed due to failed AES-256 Key Wrap decryption.

        - If the AES-256 Key Wrap decryption is successful, then the D/G M2M Node shall exit the procedure returning Kmc and the other M2M Service Connection parameters in the connectionParamSetExecute.

7.5 IVal security attributes in connection establishment

If the M2M Service Provider supports Integrity Validation (IVal), the Network M2M Node shall retrieve the IVal security attributes of the connecting D/G M2M Node from the MAS during M2M Service Connection establishment. The IVal security attributes of the D/G M2M Node shall include IValCapability and an IValKey.

NOTE 1: The interface between the Network M2M Node and MAS is not included in the present document, but it is expected that the IVal security attributes will be confidentiality and integrity protected. For purposes of specification completeness, an Authentication, Authorization and Accounting (AAA) protocol such as RADIUS or DIAMETER can be assumed.

NOTE 2: Both RADIUS [78] and DIAMETER [i.6] protocols will require new AVPs to support the inclusion of IVal security attributes.

8 M2M Secure Communication over mId

8.1 Access Network Based Security

If the M2M deployment relies on the access network security, then Kmc is not needed for mId security. The mId procedures defined in the present document may be used without any cryptographic protection when they rely on mechanisms for security not include in the present document.

8.2 Channel Security

8.2.1 Supported Channel Security Methods

The Channel-TLS-Kmc method secures TCP payloads using TLS in RFC 4346 [28] or RFC 5246 [35] with the TLS handshake conforming to TLS-PSK in RFC 4279 [27].

The Channel-DTLS-Kmc method secures UDP payloads using DTLS in RFC 6347 [72] with the TLS handshake conforming to TLS-PSK as per RFC 4279 [27].

NOTE: All normative references for this mId Security method are provided in clause 8.2.1.2.

8.2.1.1 Negotiation to use a Channel Security Method

NOTE: Since the M2M Service Connection procedure using GBA (clause 7.2.1) incorporates TLS-PSK as specified in TS 133 222 [7] for mId security, Channel-TLS method is not applicable in this case.

A Channel Security method shall be used only if it is negotiated as part of either:

- an M2M Service Connection procedure using EAP over PANA (clause 7.3); or
- an M2M Service Connection procedure using TLS-PSK (clause 7.4).

8.2.1.2 Supported TLS/DTLS Versions and TLS Cipher Suites for Channel Security Methods

The bulk encryption algorithm for TLS shall provide 128-bit security.

For a D/G M2M Node to support the Channel-TLS-Kmc Method, the D/G M2M Node shall support one of the following combinations of TLS versions and TLS ciphersuites:

- TLS v1.1 (RFC 4346 [28]) with TLS-PSK handshake (RFC 4279 [27]) and TLS Extensions specification (RFC 4366 [30]), with the following TLS ciphersuite:
    - TLS_PSK_WITH_AES_128_CBC_SHA (RFC 4279 [27]).
- TLS v1.2 (RFC 5246 [35]) with TLS-PSK handshake (RFC 4279 [27]) and TLS Extensions specification (RFC 6066 [41]), with the following TLS ciphersuites:
    - TLS_PSK_WITH_AES_128_CCM [74].
    - TLS_PSK_WITH_AES_128_CCM_8 [74].

For a Network M2M Node to support the Channel-TLS-Kmc method, the Network M2M Node needs to support all combinations of TLS versions and TLS cipher suites that are supported by the D/G M2M Nodes with which the Network M2M Node wishes to perform this mId Security method.

For a M2M Node to support the Channel-DTLS-Kmc method, the M2M Node shall support the following combination of DTLS versions and TLS ciphersuites:

- DTLS v1.2 (RFC 6347 [72] with TLS-PSK handshake (RFC 4279 [27]) and TLS Extensions specification (RFC 4366 [30]), with the following TLS ciphersuites:
    - TLS_PSK_WITH_AES_128_CCM [74].
    - TLS_PSK_WITH_AES_128_CCM_8 [74].

8.2.1.3 Details of the DTLS/TLS Handshake

8.2.1.3.1 Applicability to DTLS and TLS

NOTE: The DTLS handshake reuses the TLS handshake messages.

The Channel-TLS-Kmc method shall conform to all subclauses of clause 8.2.1.3.

The Channel-DTLS-Kmc method shall conform to all subclauses of clause 8.2.1.3.

8.2.1.3.2 TLS ClientHello.server_name Field Details For Channel Security Methods

The D/G M2M Node shall include the TLS Extension server_name field (see references to TLS Extensions specifications in clause 8.2.1.2) in the TLS ClientHello Message, and the server_name field shall include the FQDN of the Network M2M Node (this FQDN is presumed to be identical to the FQDN part of the NSCL URI).

The Network M2M Node shall process the TLS Extension ClientHello.server_name field as per TLS Extensions specifications in clause 8.2.1.2.

8.2.1.3.3 TLS ServerKeyExchange.psk_identity_hint Field Details For Channel Security Methods

NOTE: If the Network M2M Node supports the M2M Service Connection procedure using GBA then the Network M2M Node adds information in TLS ServerKeyExchange.psk_identity_hint field (see clause 7.2.1).

If the D/G M2M Node negotiated to use a Channel Security method, then the D/G M2M Node shall ignore any values in the TLS-PSK psk_identity_hint field [27] of the TLS Handshake ServerKeyExchange message.

8.2.1.3.4 TLS ClientKeyExchange.psk_identity and PSK Derivation for Channel Security Methods

The psk_identity shall be formed by the D/G M2M Node as follows:

```
psk_identity =   "ETSI M2M Rel " | <version number> | " Kmc " |
                 <channel security method name>  | " " |
                 <encoded M2M-Connection-ID> | " " |
                 <encoded Kmc-Index>
```

where:

- <version number > = <X> | '.' | <Y> where:
  - <X> is the major version number represented in decimal digits; and
  - <Y> is the minor version number represented in decimal digits.
- <channel security method name> is defined to be:
  - "Channel-TLS" if using the Channel-TLS-Kmc method; or
  - "Channel-DTLS" if using the Channel-DTLS-Kmc method.
- <encoded M2M-Connection-ID> is the 64-bit M2M-Connection-ID encoded in HexBinary type (clause 11.2).
- <encoded Kmc-Index> is the 32-bit Kmc-Index encoded in HexBinary type (clause11.2).

The (256-bit) PSK shall be generated as:

- PSK = HMAC-256(Kmc, TLS ClientKeyExchange.psk_identity).

The Network M2M Node shall terminate the DTLS session (if using Channel-DTLS-Kmc) or TLS session (if using Channel-TLS-Kmc) using a TLS "decrypt error" alert (as discussed at the end of clause 2 of RFC 4279 [27]) if any of the following scenarios occur:

- The Network M2M Node is unable to parse the psk_identity.
- The identified Kmc is not valid.
- Channel security had not been negotiated for use with the identified combination of M2M-Connection-ID and Kmc-Index.

8.3 Object Security

8.3.1 Securing CoAP-based mId

Securing CoAP using object security is discussed in annex H.

8.3.2 Securing XML-based mId

If the mId interface is using text-based or binary XML only (i.e. application/xml, application/exi, application/fastinfoset), and an object security mechanism is desired, XML signatures [61] and/or XML encryption v1.1 XML-ENCv1.1 [73] shall be used. If the mId interface is used for content types other than XML (e.g. JSON), then using XML security does not provide a complete security solution for mId as it does not apply to non-XML-based mId content types. In such cases the deployment shall not rely on object security, and instead channel and/or access network security shall be used. See annexes C and D for the listing of attributes that are carried in XML encoding.

The present document identifies the keys and key names to be used with the XML security. Other details are already governed by the [61] and XML-ENCv1.1 [73] specifications.

This mechanism is optional to implement and use. When it is used, it shall apply to the entire XML payload unless a selection is made via mechanism not included in the present document.

XML security requires that, in scenarios where both [61] and XML-ENCv1.1 [73] are applied, then separate integrity (data origin, integrity and replay protection) and encryption keys are used. Following formulas shall be used for generating those keys from Kmc.

Kimc and Kcmc are the respective integrity and encryption keys based on the Kmc.

- Kimc = HMAC-SHA256(Kmc, "Kmc Integrity Key").
- Kcmc = HMAC-SHA256(Kmc, "Kmc Encryption Key").

The Kimc shall be used as the integrity key with [61] and Kcmc shall be used as the encryption key with XML-ENCv1.1 [73]. The KeyName for Kimc shall be set according to following formula:

KeyName = "ETSI M2M Kimc Integrity " |

<Value of Connection-ID in printable string form> | " " |

<Value of Kmc index in printable string form>

EXAMPLE 1: "ETSI M2M Kimc Integrity 1234567890 9999 33".

The KeyName for Kcmc shall be set according to following formula:

KeyName = "ETSI M2M Kcmc Encryption " |

<Value of Connection-ID in printable string form> | " " |

<Value of Kmc index in printable string form>

EXAMPLE 2: "ETSI M2M Kcmc Encryption 1234567890 9999 33".

9 Resources

In stage 3 each sub-resource of multiplicity 1 is modelled as an attribute containing a reference to the sub-resource, i.e. containing its resource URI of the sub-resource. Each sub-resource of multiplicity "0..unbounded" is represented as one collection attribute that contains a list of references to the resource URIs these child resources.
Each reference in this list has an id that corresponds to the unique id of the resource in the collection.

Resources that are indicated with "<" and ">" in TS 102 690 [2], clause 6, have an identity. This identity is an explicit part of the resource representation in the form of an id attribute. The exception to this rule is the <sclBase> resource, which does not have an id attribute.

The attribute type and mandatory/optional indication from stage 2 [2] tables are mapped to 3 different columns that shall specify how the presence of an attribute in the resource representation is handled in CREATE and UPDATE requests, and if the response shall contain the attribute in the resource representation, in case such a resource representation is present.

The handling of attributes that are present or absent in a resource representation in a CREATE or UPDATE request are described in common operations (clauses 10.3.2.3, 10.3.2.4, 10.3.2.10 and 10.3.2.15).

Some resources can be announced, which means that the resource can be discovered in another SCL as the SCL where the resource is created. Resources that can be announced shall have an *announceTo* attribute. The announcing (creation of an announced resource), the update of the announced resource and the deannouncing (the deletion of the announced resource) are described in the common operations "announce resource" and "deannounce resource" and are invoked from the primitives related to resources with the *announceTo* attribute.

Some resources can be subscribed to, which means that the subscriber is notified about modifications of the subscribed-to resource. Resources that can be subscribed-to shall have a *subscriptionsReference* attribute. The subscription resource and related primitives as described in clause 10.25. Issuers that are not server capable or not publically addressable may use a notification channel in order to receive notifications. The procedures for establishing a notification channel and how to use such a channel for long polling are described in clause 10.37.

Resources can be addressed as a whole, but it is also possible to address a specific attribute of a resource. This is described in clause 10.39.

10 SCL Primitives

10.1 Introduction

The structure of the clause is as follows:

- Clause 10.3 describes the common operations of the Create, Retrieve, Update and Delete request and response primitives. These common operations are referenced from the primitive specific handling as described in the primitive specific clauses below.
- Clauses 10.4 to 10.39 describe each of the primitives as applied to the resources. The resource related to the primitives is defined in the same clause.

10.2 General aspects

10.2.1 SCL primitives

M2M shall support mIa, dIa and mId, i.e. the support of the API on mIa, dIa and mId as described in the present document is mandatory.

The SCL primitives shall apply to the M2M REST interfaces; dIa, mIa and mId. The applicability of per specific primitive is defined in the primitive definition clauses.

Each primitive is designed to map on HTTP, or CoAP, but it is described independently from HTTP and CoAP to allow other potential bindings. The primitives are designed to also support the case of an API internal to the M2M nodes, where the API shall respect the primitive description but may be bound on other communication means that are product specific (for example, to library calls). This is primarily the case of dIa, when it is internal to the node or has to map on specific capillary network protocols (D' case as described in TS 102 690 [2]), but it may apply also to mIa.

SCL primitives shall be atomic. When two concurrent primitives affect the same resource directly, one primitive shall finish completely before the second one starts. This ensures, for example, that a primitive execution can validate the existence of a resource, and then in later execution of that primitive be assured that the resource was not removed by another primitive executing at the same time.

10.2.2 Asynchronous and semi-asynchronous processing

The primitives in the following clauses are described in the synchronous processing model, i.e. a RequestIndication is received by the SCL, the receiving SCL performs some actions and sends a ResponseConfirm which indicates either success or error.

Although this is the typical interaction, there are cases where the receiving SCL may decide, for various reasons, to postpone the sending of the response. The main reason for such a decision would be if the generation of the response takes too much time, while in some transport protocols (e.g. HTTP) the client will only wait for a short time for a synchronous response. In order to handle these situations, two alternative interaction scenarios are defined; semi-asynchronous and asynchronous communication.

This clause describes these modes of communication. They are not described per primitive, but only in this clause. Some of the primitives may give hints at when the receiving SCL may decide to switch communication mode to semi-asynchronous or asynchronous, but ultimately it is a decision by the receiving SCL whether to return an immediate response, or delay the response until a later time.

On the primitive level, the issuer is not aware of the underlying mechanism that is being used, be it semi-asynchronous or asynchronous, however, it shall be informed using the STATUS_ACCEPTED *statusCode* that there eventually will be another (final) response returned to the issuer, i.e. from the issuer point of view there shall be either:

- a single ResponseConfirm to a RequestIndication; or
- two ResponseConfirms to one RequestIndication (i.e. first a response with STATUS_ACCEPTED and the later a final response).

Likewise, on the primitive level, the receiver is not aware of the underlying mechanism, be it semi-asynchronous or asynchronous, however, it shall inform the transport layer when a response will take longer by sending an extra response with a STATUS_ACCEPTED, before eventually sending the final response.

It is the transport layer that shall handle the differences between asynchronous, semi-asynchronous and synchronous communication, as is shown in annex C for HTTP and annex D for CoAP.

10.3 Common operations

10.3.1 Issuer actions

10.3.1.1 Compose RequestIndication primitive

The issuer shall compose a RequestIndication message that shall be mapped to a specific protocol.

In case the request issuer is a DA, GA or NA and the request is targeting resource(s) not hosted on the local SCL, then the RequestIndication may contain the following additional attributes for store-and-forward (SAF) handling as defined in clause 9.3.1.5 of TS 102 690 [2].

Table 10.1: SAF related primitive attributes

Additional SAF attributes of RequestIndication Primitives		
Primitive Attribute	**Mandatory/Optional**	**Description**
TRPDT (Tolerable Request Processing Delay Time)	O	The indication of TRPDT allows the local SCL that receives the request to delay forwarding of the request to the hosting SCL up to the time indicated by TRPDT - either as an absolute time or as a delay period - at the benefit of possibly aggregating more requests to the same hosting SCL
RCAT (Request CATegory)	O	The indication of RCAT allows the local SCL to block forwarding of requests to access remotely hosted resources if the SCL cannot establish connectivity using an appropriate access network for the given RCAT value, the given issuer, and (in case of NSCL) the given destination

The primitive attributes shall not be present on primitives transported on the mId interface.

The issuer shall follow table 10.1 for the RequestIndication for the specific primitive it wants to send. When including a resource representation in the request indication for create and update, the issuer shall take into account the validation rules as specified in "Check validity for resource representation for create" and "Check validity for resource representation for update", respectively. For example, any attributes marked with NP shall not be present in the resource representation for the corresponding request indication.

10.3.1.2 Send a RequestIndication to the Receiver SCL

10.3.1.2.1 Determination of the Receiver SCL

The issuer shall determine the receiver SCL.

If the issuer is an application, a DSCL registered to an NSCL or a GSCL registered to an NSCL, then receiver SCL shall be the local SCL, i.e. the SCL where the issuer is registered. If the issuer is an NSCL, the receiver SCL is determined based on the targetID as follows; the NSCL shall select the <*scl*> resource in the `<nsclBase>/scls` collection that has the following characteristics; the *link* attribute in the <*scl*> resource is a prefix of the *targetID.*

If this results in no matching resources, then the request is rejected with a STATUS_NOT_FOUND.
If results in multiple matches, then the request is rejected with a status STATUS_INTERNAL_SERVER_ERROR, since this shall never happen.

Once the receiver SCL is determined, the issuer shall select the communication channel to use for communicating with that SCL. The selection of the communication channel is described in the next clause.

10.3.1.2.2 Selection of communication channel

10.3.1.2.2.1 Issuer is an M2M Application

In case the issuer is an Application the selection of the communication channel and the routing towards the local SCL is not included in the present document:

- A DA on a D-type M2M Device or a GA on M2M Gateway is expected to be co-located with the DSCL and GSCL respectively and will use some internal mechanism, like a library call that is hard-wired to a specific SCL instance, or an internal bus structure.

- A D'A on a D'-type M2M Device is expected to use some local area network to reach its GSCL.

- An NA will most likely do a DNS lookup on a the FQDN part of the URI of the NSCL's sclBase resource that is provided to it using some offline means, and then route the request correspondingly.

10.3.1.2.2.2 Issuer is a DSCL or GSCL

For the case that the issuer is a DSCL or GSCL, the access network selection/communication channel selection is described in this clause, in line with the store and forward handling defined in clause 9.3.1.5 of TS 102 690 [2], and based on the configured policies for SAF (store-and-forward) handling that are also described in clause 9.3.1.5 of TS 102 690 [2].

The procedure for the D/GSCL to send a request to an external receiver SCL shall be as follows:

1) **Select the applicable SAF policy set:** If there are specific SAF policy sets for the requesting entity provisioned to the D/GSCL and if one of the requesting-entity-specific SAF policy sets matches with the ID of the requesting entity (e.g. in case etsiSclMo is used to provision policies, the *policyScope* attribute of a '<*safPolicySet*>' resource matches with the APP-ID of the requesting entity), the D/GSCL shall use that set of policies in the steps below. If none of the requesting-entity--specific SAF policies provisioned to the D/GSCL matches with the ID of the requesting entity, the provisioned default SAF policy set shall be applied. If no default policy set is provisioned to the D/GSCL, the request shall be rejected with a STATUS_FORBIDDEN. And the operation ends.

2) **Determine if an appropriate communications channel is already available:** After selecting the receiver SCL as described in clause 10.3.1.2.1, the D/GSCL shall determine an ordered list of access networks candidates that may already provide an active communication channel to the remote SCL. The D/GSCL shall use the RCAT (Request CATegory) of the received request to compose an ordered list of preferred access networks as defined in the policy for 'Selection among appropriate access networks' (see item 3 in clause 9.3.1.5.6.2 in TS 102 690 [2], see also representation of that policy in the '*anSelList'* attribute of the m2mSpPolicy resource of the etsiSclMo defined in clause E.2.1.4 in TS 102 690 [2]). Only access networks that are already in use for an active communication channel are added to the list, while keeping their order. The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.4 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present document or by out-of-scope mechanisms.

3) The D/GSCL shall determine which access networks out of the candidate list produced in step 2 are appropriate to use for the given RCAT value of the issued request. The D/GSCL shall do the following for each item of the list produced in step 2 - in the order they appear in the list:

   a) Verify whether the respective access network would currently be appropriate to use for the given RCAT value of the issued request by checking the policy 'Schedule of RCAT values versus time' for the respective access network (see item 1 in clause 9.3.1.5.6.1 in TS 102 690 [2], see also representation of that policy in the '*rcatSchedule'* resource of the etsiSclMo defined in clause E.2.1.3 of TS 102 690 [2]). The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.3 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms.

   b) If the D/GSCL detects that this access network is marked as appropriate to use for the given RCAT value and the current time, then the D/GSCL shall select this access network and shall try to send the composed RequestIndication over the already established communication channel on that access network to the receiver SCL:

      i) If the attempt was successful the action 'Send a RequestIndication to the Receiver SCL' is terminated.

      ii) If the attempt was not successful, the D/GSCL shall continue with the next list item.

   c) When the last list item has been processed, continue with the step 4 below.

4) **No appropriate communication channel is available:** I.e. none of the already active communication channel was determined to be appropriate for use with the RCAT value of the issued request. The D/GSCL shall check the TRPDT (Tolerable Request Processing Delay Time) on the request.

5) If the TRPDT of the issued request indicates that the request shall be forwarded without additional delay, then continue with step 10.

6) If the TRPDT value of the issued request allows for additional delay by the D/GSCL, then the D/GSCL may buffer the request in a buffer specific for the given RCAT value. If and when the D/GSCL decides to buffer a request is governed by SAF policies as described in clause 9.3.1.5.6.2 in TS 102 690 [2], see also representation of these policies in the m2mSpPolicy resource of the etsiSclMo defined in clause E2.1.4 of TS 102 690 [2]. The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.4 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms. The D/GSCL may decide not to buffer a request, e.g. because it may limit the maximum size of the RCAT specific buffers, in both number of requests or total number of buffered bytes. If the D/GSCL decides not to buffer the request, then the sequence continues with step 10.

7) If the D/GSCL decides to buffer the issued request (e.g. if the RCAT specific buffer has not yet reached its maximum) then the D/GSCL shall start a timer to monitor when the maximum allowed additional delay indicated by the TRPDT value for the issued request expires. The timer may be shorter than the delay indicated by the TRPDT value, but shall not exceed the delay indicated by the TRPDT value.

8) The D/GSCL shall send a STATUS_ACCEPTED response to the issuer. The operation sequence is stopped and the D/GSCL waits for:

   a) either a communication channel to become available on which the issued request can be sent (i.e. which is appropriate to use for traffic generated by a request of the RCAT value as indicated in the issued request): In that case the sequence continues with step 13; or

   b) until the TRPDT timer expires: In that case the sequence continues with step 9.

9) If the TRPDT timer has expired then execute step 10, using the RCAT value of the request whose timer expired.

10) **Attempt to establish a communication channel for sending pending request of a given RCAT value:** This step happens when either the TRPDT timer for a request expires, when the D/GSCL decides not to buffer a request, or when a request with a TRPDT timer of 0 was issued. This is done in two steps:

   a) The D/GSCL shall produce an ordered list of access networks candidates. This list shall be composed by the D/GSCL by looking up the preferred access networks for the given RCAT value of the issued request as defined in the policy for 'Selection among appropriate access networks' (see item 3 in clause 9.3.1.5.6.2 in TS 102 690 [2], see also representation of that policy in the '*anSelList'* attribute of the m2mSpPolicy resource of the etsiSclMo defined in clause E.2.1.4 in TS 102 690 [2]). The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.4 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms.

   b) Then the D/GSCL shall verify whether the items in the list of candidate access networks would currently be appropriate to use for the given RCAT value by checking the policy 'Schedule of RCAT values versus time' for the respective access network (see item 1 in clause 9.3.1.5.6.1 in TS 102 690 [2], see also representation of that policy in the '*rcatSchedule'* resource of the etsiSclMo defined in clause E.2.1.3 of TS 102 690 [2]). The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.3 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms. Candidate access networks that are not deemed appropriate shall be removed from the list.

   The end result of this step is an ordered set of candidate access networks that can be used to make attempts to establish communication channels for sending the pending requests of the given RCAT value.

11) If the resulting list of candidate access networks is empty then actions of the D/GSCL depend on whether the request that triggered the attempt was a buffered request or not.

   a) If the request was not a buffered request (i.e. it is not a timeout of the TRPDT that triggered step10), then the request shall be rejected with a STATUS_GATEWAY_TIMEOUT.

   b) If the request was a buffered request (i.e. step 10 was triggered by a timeout of the TRPDT timer), then the D/GSCL shall send an semi-asynchronous or asynchronous response, with a STATUS_GATEWAY_TIMEOUT. The request is removed from the buffer.

   In both cases the operation procedure ends.

12) The D/GSCL shall try to establish a communication channel to the receiver SCL using the first element in the list of candidate access networks:

   a) If that attempt is not successful, the D/GSCL shall not attempt to establish a communication channel using the same candidate access network for a time governed by policies set by the Access Network Provider as defined in item 2 in clause 9.3.1.5.6.1 in TS 102 690 [2], see also representation of that policy in the '*blockPeriods'* resource of the etsiSclMo defined in clause E.2.1.3 of TS 102 690 [2]. The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.3 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms. The D/GSCL shall remove the first item from the list of candidate access networks and continue with step 11.

   b) If the attempt was successful, continue the sequence with the next step.

13) **New communication channel established / changes in property of being appropriate access network:** When a new communication channel is established or when an access network that provides an already active communication channel has changed its property of being appropriate to use for given RCAT values (e.g. a time period has started during which requests of a certain RCAT value may be send over an existing communication channel), the D/GSCL shall perform the following steps.

14) The D/GSCL shall lookup all RCAT values for which the use of the currently available communication channel(s) is appropriate.

15) For each RCAT value in this list the D/GSCL shall:

    a) Find all buffered requests that are pending for that RCAT value.

    b) Send all identified requests of that RCAT value to the receiver SCL using one of the communication channels provided by an appropriate access network for that RCAT value and delete the corresponding TRPDT timers. If more than one communication channel is available for the specific RCAT value, the order as defined in the policy shall be used to select the most appropriate communication channel. The policy that is applied here is 'Selection among appropriate access networks' (see item 3 in clause 9.3.1.5.6.2 in TS 102 690 [2], see also representation of that policy in the '*anSelList'* attribute of the m2mSpPolicy resource of the etsiSclMo defined in clause E.2.1.4 in TS 102 690 [2]). The provisioning of these policies is either carried out by means of REM procedures using the m2mSpPolicy resource of the etsiSclMo, see clause E.2.1.4 of TS 102 690 [2] and corresponding primitives for <*mgmtObj*> resources in the present TS or by out-of-scope mechanisms.

10.3.1.2.2.3 Issuer is an NSCL

For an NSCL, the selection of the appropriate access network and communication channel is described here in more detail.

The communication channel between a DSCL/GSCL and an NSCL is typically setup from the DSCL/GSCL towards the NSCL. The NSCL may also trigger the DSCL/GSCL to establish such a channel, using a wakeup procedure, but this procedure is not defined by the present document.

A communication channel may be used both ways, or it may be one way (e.g. only from D/GSCL to NSCL). The latter typically is the case if the DSCL or GSCL is not server capable or is not addressable from the NSCL due to NAT and firewall related issues. Theoretically, the communication channel could also be one way in the other direction, i.e. from NSCL to D/GSCL. This is treated in the same way as a both-way communication channel in the rest of this text.

An active communication channel that is established and that can potentially be used by the NSCL to reach the DSCL or GSCL, is identified by a m2mPoc resource. The m2mPoC may be populated with the contactURI of a notificationChannel, for instance in the case the SCL is server capable but resides behind a NAT function.

In addition to using an active communication channel as identified by an m2mPoc resource, the NSCL may also reach a DSCL or GSCL for notification via a notificationChannel. This method is described in more detail in clause 10.37.7.2.

The procedure for the NSCL to send a request to an external receiver SCL shall be as follows:

1) After selecting the receiver SCL as described above and identifying the corresponding <scl> resource, the NSCL shall retrieve the m2mPocs collection child resource of that <scl> resource.

2) If that collection is empty, no communication channel is available to reach the receiver SCL. The handling then shall be as described below in step 9.

3) If the m2mPocs collection child of the <scl> resource is not empty, then the NSCL shall select the active m2mPoC resources from the collection, this are those m2mPoC resources whose *onlineStatus* is marked as ONLINE or NOT_REACHABLE.

4) To reduce and order the set of selected m2mPoc resources, the NSCL shall perform access network selection between these active m2mPocs.
   Network selection is a two part process. Network selection is based on policy data in the NSCL under control of the service provider and policy data under control of the network access operator. How this data is provisioned is not included in the present document. The minimal policies are described in clause 9.3.1.5 of TS 102 690 [2]:

   a) **Select the applicable SAF policy set:**

      1) If the consolidated SAF policies contain a set of policies for the combination of requestingEntity and destination D/GSCL ID, then the NSCL shall use that set of policies in the steps below, otherwise.

      2) If the consolidated SAF policies contain a set of policies for the destination D/GSCL ID, then the NSCL shall use that set of policies in the steps below, otherwise.

      3) If the consolidated SAF policies contain a set of policies that are NSCL-wide, then the NSCL shall use that set of policies in the steps below, otherwise.

      4) The request shall be rejected with a STATUS_FORBIDDEN.

   b) The first part of the network selection shall be controlled by the service provider policies and shall be based on the RCAT that is associated with the request. The RCAT shall be mapped to specific access networks, and access networks are associated with active m2mPoc resources. The *onlineStatus* of the m2mPoC also may serve as a criterion in the selection and ordering process, e.g. m2mPoc resources marked as NOT_REACHABLE may get lower priority compared to m2mPoc resources with an *onlineStatus* marked as ONLINE.
      The result of this step shall be an ordered set of active m2mPocs which according to SP policies are appropriate for the request with this RCAT in these circumstances. The corresponding policies are defined in clause 9.3.1.5.6.2 in TS 102 690 [2], but the provisioning of these policies is not included in the present document.

   c) The second part of the network selection process shall be controlled by the configuration provided by the access network provider. Each m2mPoc is associated with an access network and for each access network corresponding policies shall apply. These policies are applied to each m2mPoc in the collection resulting from step b; This step may exclude an m2mPocs depending on time, based on schedule information. Optionally, other information may be taken as input in this selection process. The corresponding policies are defined in clause 9.3.1.5.6.1 in TS 102 690 [2] but the provisioning of these policies is not included in the present document.

   d) The final result shall be an ordered set of m2mPoc resources, to be tried in order of preference.

5) If the resulting reduced set is empty, the handling shall be as described below in step 9.

6) The NSCL shall try to send the composed RequestIndication over the first m2mPoc in the resulting set:

   a) In order to deliver a request using an m2mPoc, the NSCL shall retrieve the *contactInfo* element from the m2mPoc resource. The *contactInfo* is a URI, of which only the schema, the host and port information shall be used for the purposes of routing.

   b) If the *contactInfo* contains ip-address based host and port information, this shall be used.

   c) If the *contactInfo* contains a FQDN in the host part, then the NSCL shall resolve the FQDN to a ip-address and port, e.g. using DNS.

   d) If the *contactInfo* contains other information, then the NSCL shall resolve this information to a host and port, using the access network provider. This option is for interoperability with current and future access networks.

   e) The NSCL may optimize this resolving process by caching the host and port information received from earlier lookups.

   f) If the *contactInfo* cannot be resolved to host and port information, the request shall be rejected with a STATUS_GATEWAY_TIMEOUT.

g) If the *contactInfo* is resolved to a host and a port, then the request shall be sent to this host and port. The schema part of the *contactInfo* shall determine what protocol is used for the communication.

7) If the request delivery failed and this is detected by the NSCL, and the corresponding m2mPoc resource is marked as ONLINE, then the corresponding NSCL shall be marked as NOT_REACHABLE, the m2mPoc is removed from the selected set and the process is repeated from step 5.

8) If the request delivery succeeded and this is detected by the NSCL, and the corresponding m2mPoc resource is marked as NOT_REACHABLE, then the corresponding NSCL shall be marked as ONLINE and this procedure ends.

9) **No communication channel available:** This step is reached if no m2mPoc could be selected which can be used to successfully send the request. The NSCL shall check the TRPDT on the request.

10) If the TRPDT is 0, then continue with step 15.

11) If the TRPDT is larger than 0, then the NSCL may buffer the request in a buffer specific for the specified RCAT. If and when the NSCL decides to buffer a request is outside of the scope of the present document. The NSCL may decide not to buffer a request, e.g. because it may limit the maximum size of the RCAT specific buffers, in both number of requests or total number of buffered bytes. The corresponding policies are defined in clause 9.3.1.5.6.2 in TS 102 690 [2], but the provisioning of these policies is not included in the present document. The NSCL may provide additional criteria on when to buffer requests.
If the NSCL decides not to buffer the request, then the request shall be sent immediately and the procedure continues with step 15.

12) If the NSCL decides to buffer the request (e.g. if the RCAT specific buffer has not yet reached its maximum) then the NSCL shall start a timer to monitor for the TRPDT expiration. The timer may be shorter than the TRPDT, but shall not exceed the TRPDT value.

13) The NSCL sends an STATUS_ACCEPTED response to the issuer. The sequence is stopped and the NSCL waits for:

a) either a communication channel to become available on which request can be sent (i.e. a channel for which the policies allow traffic for the RCAT indicated in the request) . In that case the sequence continues with step 21; or

b) until the TRPDT timer expires. In that case the sequence continues with step 14.

14) If the TRPDT timer expires then execute step 15, with the request whose timer expired.

15) **Attempt to establish a communication channel for an RCAT:** This step is reached when either:

a) the TRPDT timer for a request expires;

b) the NSCL decides not to buffer a request; or

c) a request with a TRPDT timer of 0 is requested.

16) The NSCL may do another access network selection, but now based on all the *potential* access networks. This means that it also takes into account m2mPoc resources with an *onlineStatus* set to OFFLINE and in addition takes into account m2mPocs that may be created by the device or gateway after a wakeup is sent.
The process again is a two step process, where first based on service provider specified policies (see clause 9.3.1.5 of TS 102 690 [2]) a number of access networks are selected in order of preference and a second step that is based on policies provided by the access network provider for each of the selected access networks. As in step 5, the policies may use more information as described in clause 9.3.1.5.6.1 in TS 102 690 [2] and as in step 5 the provisioning of said policies is not included in the present document.
In addition, the NSCL shall also consider the blocking policies mentioned in step 21. I.e. access networks which are still blocked based on a previous wakeup attempt shall not be retried until the blocking timer has expired.
The end result of this step is an ordered set of potential access networks that can be used to wakeup a device. It is possible that the NSCL does not have any possibility or desire to wakeup the device in which case the set will be empty. Note that wakeup of a device is not specified as a normative procedure in the present document.

17) If the resulting set is empty then actions of the NSCL depend on whether the request that triggered the establishment was a buffered request or not:

    a) If the request was not a buffered request (i.e. it is not a timeout of the TRPDT that triggered step 18), then the request is rejected with a STATUS_GATEWAY_TIMEOUT.

    b) If the request was a buffered request (i.e. step 18 was triggered by a timeout of the TRPDT timer), then the NSCL shall send an a semi-asynchronous or synchronous response, with a STATUS_GATEWAY_TIMEOUT.

18) The NSCL may send a wakeup request to the device belonging to the receiver SCL. The wakeup request indicates the preferred communication channel establishment parameters based on the first element in the set of potential access networks.

19) The receiver SCL may establish a communication channel based on the received wakeup request. Such an incoming channel establishment is handled in step 21.

20) If the receiver SCL did not establish a communication channel, the NSCL may remove the first item from the list and continue with step 17. The NSCL shall not attempt to wakeup the device using the same access network for a time governed by policies set by the Access Network Provider. These policies are defined in clause 9.3.1.5 of TS 102 690 [2], but the provisioning to the NSCL is not included in the present document.

New communication channel established/changes in property of being appropriate access network:

21) Two events trigger the following sequence of actions

    a) a new incoming communication channel is established to the NSCL. In this case the DSCL/GSCL shall inform the NSCL about this by either creating a new m2mPoc resource with an *onlineStatus* set to ONLINE or by updating the *onlineStatus* of existing m2mPoc from OFFLINE to ONLINE; or

    b) if there is any change in the criteria that govern the network selection (e.g. a time period has started during which traffic of a certain RCAT may be send over an existing communication channel) the NSCL shall perform the following steps.

22) The NSCL shall locate all the RCATs that can be handled by the access network connection represented by this m2mPoc resource according to the consolidated policies.

23) For each RCAT in this list the NSCL shall find all buffered requests for this RCAT for this receiver SCL.

24) For each request in this buffer, the NSCL shall send the request to the receiver SCL using this connection.

25) Once a request is sent, the corresponding timer based on the TRPDT shall be stopped.

10.3.1.3 Wait for ResponseConfirm primitive

The issuer shall wait for the a ResponseConfirm primitive from the receiver that corresponds to the RequestIndication primitive that was sent by the issuer. Correlation between the RequestIndication and the corresponding ResponseConfirm is handled by the transport layer.

If no ResponseConfirm primitive is received within a certain time, specified by server policy and/or by the underlying transport technology, this shall be handled as if a ResponseConfirm primitive with a statusCode STATUS_REQUEST_TIMEOUT was received.

10.3.2 Hosting SCL actions

10.3.2.1 Check existence of the addressed resource

The hosting SCL shall check if the resource addressed by the targetID exists in the repository. If the resource does not exist, the hosting SCL shall reject the request with a "STATUS_NOT_FOUND".

10.3.2.2 Check the syntax of received message

Message validity is specified by the protocol mapping annexes. Annex C and by validating the received resource representation (e.g. in plain XML, binary XML or JSON) against the provided schema definitions in annex B.

If the message is not valid, the request shall be rejected with a "STATUS_BAD_REQUEST".

10.3.2.3 Check validity of resource representation for CREATE

The handling below shall apply t to each attribute in the resource for CREATE request primitives and the handling depends on the "presence in CREATE request" column of the resource table. If the request is rejected based on the rules below, then the other attributes do not have to be checked.

If no resource representation is present in the CREATE request, then the request is rejected with a STATUS_BAD_REQUEST statusCode.

The id attribute has special handling. If the id-attribute is present in the CREATE request, the hosting SCL shall check if a resource with the same id already exists in the addressed collection. If such a resource exists and the response column is marked as M, then the hosting SCL shall reject the request with a "STATUS_CONFLICT".

If the expirationTime attribute is present in the resource representation, but its value indicates a time in the past, then the request shall be rejected with a STATUS_BAD_REQUEST.

N/A attribute

Indicates that the CREATE request is not supported .This common procedure is never referenced from a CREATE request.

M attribute

If the attribute is present in the resource representation in the CREATE request, the hosting SCL shall check if the value is acceptable according to internal policies.

If the provided value is not accepted and the response column is marked M then the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

If the attribute is not present in the resource representation in the CREATE request the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

O attribute

If the attribute is present in the resource representation in the CREATE request, the hosting SCL shall check if the value is acceptable according to internal policies.

If the provided value is not accepted and the response column is marked M or O then the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

NP attribute

If the attribute is present in the resource representation in the CREATE request, the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

10.3.2.4 Check validity of resource representation for UPDATE

The handling below shall apply to each attribute in the resource for UPDATE request primitives and the handling depends on the "presence in UPDATE request" column of the resource table. If the request is rejected based on the rules below, then the other attributes do not have to be checked.

If the expirationTime attribute is present in the resource representation, but its value indicates a time in the past, then the request shall be rejected with a STATUS_BAD_REQUEST.

N/A attribute

Indicates that the UPDATE request is not supported .This common procedure is never referenced from a UPDATE request.

M attribute

If the attribute is present in the resource representation in the UPDATE request, the hosting SCL shall check if the value is acceptable according to internal policies.

If the provided value is not accepted and the response column is marked M, the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

If the attribute is not present in the resource representation in the UPDATE request, the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST".

O attribute

If the attribute is present in the resource representation in the UPDATE request, the hosting SCL shall check if the value is acceptable according to internal policies.

If the provided value is not accepted and the response column is marked M or O then the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST" statusCode.

NP attribute

If the attribute is present in the resource representation in the UPDATE request, the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST" unless the value provided for the attribute exactly matches the value in the current resource representation stored in the hosting SCL. In addition, the lastModifiedTime attribute shall always be accepted (but ignored) by the hosting SCL, no matter what value was provided in the request.

10.3.2.5 Check authorization of the requestingEntity based on accessRightID

Check if the requestingEntity is authorized for the requested action. The authorization is based on the accessRightID attribute of the resource addressed in the targetID.

The accessRight resource whose resource URI matches the accessRightID shall be retrieved using the accessRightRetrieveRequestIndication primitive.

In case the accessRight resource is located on an SCL that is not the hosting SCL, the hosting SCL may have cached a representation of that accessRight resource, and if so, the hosting SCL shall use the cached representation. In such a case the hosting SCL shall keep the cached representation inline with the representation of the accessRight resource living on the remote SCL.

If the accessRight resource is retrieved successfully, the access shall be granted if one of the following applies:

- **RetrieveRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the permissions attribute, whose permissionFlag includes the READ permission flag.
- **UpdateRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the permissions attribute, whose permissionFlag includes the WRITE permission flag.
- **DeleteRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the permissions attribute, whose permissionFlag includes the DELETE permission flag.
- **CreateRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the permissions attribute, whose permissionFlag includes the CREATE permission flag.

A requestingEntity shall be considered to be one of the permissionHolders if any of the following conditions is true:

- The permissionHolders contains an <all/> element.
- Its URI matches one of the holderRefs.

- One of the domains in the permissionHolder is a prefix of its URI.
- Its URI is a member of some group resource whose URI matches one of the holderRefs. Member is here defined recursively (i.e. a member of a sub-group is also considered a member of the group).
- Its corresponding ID matches one of the applicationIDs in the applicationIDs set in the permissionHolders, in case the URI belongs to an application resource.
- Its corresponding ID matches one of the sclIDs in the sclIDs sets in the permissionHolders, in case the URI belongs to an scl resource.
- Its corresponding <sclBase> URI matches on the holderRefs in case the URI belongs to an SCL.
- Its corresponding <sclBase> URI is a member of some group resource defined as above

Here "corresponding ID" is defined as follows:

- For an <scl> URI, the corresponding ID is defined as; the id attribute of the <scl> resource on the hosting SCL (where the SCL is registered), or the path element of the resource URI after the last 'scls' in the URI.
- For an application URI, the corresponding ID is defined as; the id attribute of the <application> resource on the SCL where the application is registered, or the part of the URI after the last 'applications' in the URI.

The "corresponding <sclBase> URI> is defined as follows: the value of the 'link' attribute in the <scl> resource.

If the resource corresponding to the accessRightID cannot be retrieved, or if the addressed resource does not have an accessRightID, then the steps as described in "check authorization of the requestingEntity based on default access rights" shall be applied.

If the requestingEntity does NOT have the permission as indicated above, then the request is rejected with a statusCode. The statusCode depends on the discovery permission:

- If the requestingEntity is one of the permissionHolders for a permission element in the permissions attribute whose permissionFlag set is not empty, (i.e. this includes a DISCOVER permission flag), then the request shall be rejected with a "STATUS_PERMISSION_DENIED".
- Otherwise the request is rejected with a "STATUS_NOT_FOUND".

The rationale behind this is that for security reasons, requestingEntities shall not be able to deduct the existence of a resource they cannot discover from the returned statusCode.

10.3.2.6 Check authorization of the requestingEntity based on selfPermission

Check if the requestingEntity, is authorized for the requested action. The authorization is based on the selfPermissions attribute of the accessRight resource.

If the accessRight resource is retrieved successfully, the following rules shall apply:

- **RetrieveRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the selfPermissions attribute whose permissionFlag includes the READ permission flag.
- **UpdateRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the selfPermissions attribute whose permissionFlag includes the WRITE permission flag.
- **DeleteRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the selfPermissions attribute whose permissionFlag includes the DELETE permission flag.
- **CreateRequestIndication:** the requestingEntity shall be one of the permissionHolders for a permission element of the selfPermissions attribute whose permissionFlag includes the CREATE permission flag.

An requestingEntity is considered to be one of the permissionHolders shall any of the following conditions is true:

- Its URI matches one of the holderRefs.
- Its URI starts with one of the domains.

- The permissionHolders contains an <all/> element.
- Its URI is a member of some group resource whose URI matches one of the holderRefs, where member is here defined recursively (i.e. a member of a sub-group is also considered a member of the group).
- Its corresponding ID matches one of the applicationIDs in the applicationIDs set in the permissionHolders, in case the URI belongs to an application resource.
- Its corresponding ID matches one of the sclIDs in the sclIDs sets in the permissionHolders, in case the URI belongs to an scl resource.
- Its corresponding <sclBase> URI matches on the holderRefs in case the URI belongs to an SCL.
- Its corresponding <sclBase> URI is a member of some group resource defined as above.

Here "corresponding ID" is defined as follows:

- For an <scl> URI, the corresponding ID is defined as; the id attribute of the <scl> resource on the hosting SCL (where the SCL is registered), or the path element of the resource URI after the last 'scls' in the URI.
- For an application URI, the corresponding ID is defined as; the id attribute of the <application> resource on the SCL where the application is registered, or the part of the URI after the last 'applications' in the URI.

The "corresponding <sclBase> URI> is defined as follows: the value of the 'link' attribute in the <scl> resource.

If the requestingEntity does NOT have the permission as indicated above, then the request is rejected with a statusCode. The statusCode depends on the discovery permission:

- If the requestingEntity is one of the permissionHolders for a permission element in the selfPermissions attribute whose permissionFlag set is not empty, (i.e. this includes a DISCOVER permission flag), then the request shall be rejected with a "STATUS_PERMISSION_DENIED".
- Otherwise the request is rejected with a "STATUS_NOT_FOUND".

The rationale behind this is that for security reasons, requestingEntities shall not be able to deduct the existence of a resource they cannot discover from the returned statusCode.

10.3.2.7 Check authorization of the requestingEntity based on default access rights

The default access permissions shall grant all permissions (i.e. the full set of permissionsFlags) to the following permission holders depending on the prefix of URI of the addressed resource; the permissionHolders for prefixes from most specific to least specific are as follows:

- *<sclBase>/scls/<scl>/applications/<applicationAnnc>*: the permissionHolders shall be the hosting SCL, the SCL corresponding to the *<scl>* resource and the Application corresponding to the *<applicationAnnc>* resource.
- *<sclBase>/scls/<scl>*: the permissionHolders shall be the hosting SCL and the SCL corresponding to the *<scl>* resource.
- *<sclBase>/applications/<application>*: the permissionHolders shall be the hosting SCL and the Application corresponding to the *<application>* resource.
- *<sclBase>*: the permissionHolder shall hosting SCL.

If the requestingEntity does not match one of the permissionHolders identified above, then the request shall be rejected with a STATUS_NOT_FOUND.

10.3.2.8 Announce resource

If an announceable resource contains the *announceTo* attribute in its representation, the actions described in clause 10.3.2.8.1 shall be applied, even if the *announceTo* attribute contains an empty list of *sclBase* resource URIs. If the *announceTo* attribute is not present in the resource representation, then the actions in clause 10.3.2.8.2 shall be performed.

10.3.2.8.1 Update of announce on request of application

This action shall apply if the following conditions are met:

- A new announceable resource is created in the local SCL and the CREATE request contains an *announceTo* attribute.
- The *announceTo* attribute of the announceable resource is added/changed.

The local SCL keeps track of the SCLs where the resource is currently announced, which corresponds to the list of *sclBase* URIs from the previous value of the *announceTo* attribute. For the purposes of this procedure, the previous version of the *announceTo* attribute is called previous-*announceTo*.

1) If the *announceTo* attribute contains the *activated* element set to FALSE and the original request that triggered this action is not a CREATE., then the local SCL shall reject the original request with STATUS_NOT_FORBIDDEN and then this procedure stops.

2) If the *announceTo* attribute contains the *activated* element set to FALSE and the original request that trigger this action is a CREATE, then the *announceTo* attribute value is stored as-is in the resource. No further actions are required.

3) If the *announceTo* attribute contains the *activated* element set to TRUE, or if the *announceTo* attribute does not contain the *activated* element, then the local SCL shall:

    a) Check if the SCLs indicated in the *sclList* element of the *announceTo* attribute are registered to/from this local SCL. If any of the SCLs in the *sclList* is not registered then those SCLs are removed from the *sclList* and no further actions for those SCLs are performed.

    b) Send *createXXXAnnouncementResourceRequestIndication* (where XXX is replaced by the type of the resource to be announced) for each SCL in the *sclLists* element of the *announceTo* attribute that is NOT yet included in the previous-*announceTo*. The request includes:

        - *searchStrings* from the original resource;
        - *accessRightID* from the original resource;
        - *link* is set to the URI of the original resource;
        - *requestingEntity* is set to the application;
        - *issuer* is set to its own SCL ID (the local SCL performing the action);
        - *id* of the resource shall be set to the *id* of the original resource postfixed with Annc. I.e. if the original resource has *id* "myApp", the announced resource shall have the *id* "myAppAnnc";
        - *expirationTime* handling is to the discretion of the SCL implementation. It is the responsibility of the local SCL to keep the announced resource in sync with the lifetime of the original resource, as long as the announcement is active. One strategy to minimize signalling would be to request the same expiration from the original resource. If this is accepted by the remote SCL, then no explicit de-announce is needed in case of expiration of the original resource;
        - *targetID* is set as follow.

Table 10.2: Mapping from original resource URI to TargetID

original resource URI	targetID
<sclBase1>/applications/<appID>	<sclBase2>/scls/<scl1>/applications/
<sclBase1>/containers/<containerID>	<sclBase2>/scls/<scl1>/containers/
<sclBase1>/groups/<groupID>	<sclBase2>/scls/<scl1>/groups/
<sclBase1>/accessRights/<accessRightID>	<sclBase2>/scls/<scl1>/accessRights/
<sclBase1>/applications/<appID>/containers/<containerID>	<sclBase2>/scls/<scl1>/applications/<appID>Annc/containers/
<sclBase1>/applications/<appID>/accessRights/<accessRightID>	<sclBase2>/scls/<scl1>/applications/<appID>Annc/accessRights/
<sclBase1>/applications/<appID>/groups/<groupID>	<sclBase2>/scls/<scl1>/applications/<appID>Annc/groups/

On table 10.2 the *sclBase1* is the URI of the *sclBase* of the announce-to SCL as indicated in the *sclList* in the *announceTo* attribute. In table 10.2 the `<sclBase2>/scls/<scl1>` is the resource that was created when the SCL represented by <*sclBase1*> registered the scl represented by <*sclBase2*>.

c) Ignore all SCLs in the *sclList* element of the *announceTo* attribute that were already included in the previous-*announceTo*.

d) Send *deleteXXXAnnouncementResourceRequestIndication* (where XXX is replaced by the type of resource to be de-announced) for each SCL in the previous-*announceTo* that is not included in the *sclList* of the provided *announceTo* attribute. The request shall include the URI of the announcement resource to be removed. The request includes:

- *requestingEntity* is set to the application;
- *issuer* is set to its own SCL ID (the local SCL performing the action);
- *targetID* is set to the resource URI of the previously announced-resource on the remote SCL. The local SCL received and stored the URI of the announced resource after it was created.

e) Waits until all the createXXXAnnouncementResourceResponseConfirm and/or deleteXXXAnnouncementResourceResponseConfirm are received and it acts as follow:

i) For each unsuccessful *createXXXAnnouncementResourceResponseIndication*, the remote SCL is removed from the *sclList* in the *announceTo* attribute.

ii) For each successful *createXXXAnnouncementResourceResponseIndication*, the local SCL shall internally store the resourceURI of the created announced resource. This URI is needed for delete the resource later on.

iii) For each unsuccessful *deleteXXXAnnouncementResourceRequestIndication* with the *statusCode* STATUS_NOT_FOUND, the remote SCL is removed from the *sclList* in the *announceTo* attribute. For all other *statusCode* value, no action is performed.

iv) For each successful *deleteXXXAnnouncementResourceRequestIndication*, the remote SCL is removed from the *sclList* in the *announceTo* attribute.

4) The local SCL shall update the *announceTo* attribute in the resource.

10.3.2.8.2 Update of announce on request of local SCL

This action shall apply if the following conditions are met:

- A new announceable resource is created in the local SCL, the CREATE request does not contain an *announceTo* attribute.
- The *announceTo* attribute is removed from the resource representation by an XXX*updateRequestIndication* (where XXX represents the resource that contains the *announceTo* attribute).

The local SCL shall determine the list of SCLs where the resource shall be announced as follows:

1) It checks if the issuer is an application registered to this local SCL, if not the procedure stops here, otherwise the local SCL checks if the application registration resource has an *announceTo* attribute whose *global* element is set to TRUE. (If *global* is set to FALSE the procedure does not apply).

2) The *sclList* of *announceTo* of the application Registration is used (note that the list may be empty, in case the issuer wants to forbid announcing any created resources).

3) If the *activate* element of the *announceTo* attribute from the application resource is set to FALSE, then the local SCL adds the *announceTo* attribute in the announceable resource (that trigger this procedure) and it sets the *activate* element to FALSE. The action returns.

4) Otherwise (i.e. *global* set to FALSE or no *announceTo* is present), the SCL determines based on its policies, if the resource shall be announced:

    a) If the policies state the resource shall not be announced, this action returns and the original procedure is continued.

    b) Otherwise, the SCL determines which SCLs shall be in the *sclList* element of the *announceTo* attribute. The SCL also set the *active* element to TRUE.

Then, the local SCL shall announce the resource to the SCLs in the sclList element. This procedure is described in step 3, from a to e in clause 10.3.2.8.1, with the exception that the *requestingEntity* for any initiated request is set to the SCL hosting the announcing resource.

5) The *announceTo* with the updated *sclList* is set on the original resource (that trigger this procedure). This may trigger notifications to resources that have subscribed to changes in the original resource or its *announceTo* attribute.

10.3.2.8.3 Create announced Resource

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" (clause 10.3.2.13) and then "Send ResponseConfirm primitive" (clause 10.3.2.17). The corresponding *statusCode,* as indicated in enumeration StatusCode in clause 11.3 shall be included in the *ResponseConfirm* primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check the existence of the addressed resource". In this case the addressed resource shall be the provided collection resource.

3) "Check authorization of the *requestingEntity* based on *accessRightID*". In this case the checked access rights are those of the addressed collection.

4) "Check the syntax of the received message".

5) "Check validity of the resource representation for CREATE".

6) "Create the resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.3.2.8.4 Retrieve announced Resource

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" (clause 10.3.2.13) and then "Send ResponseConfirm primitive" (clause 10.3.2.17). The corresponding *statusCode,* as indicated in enumeration StatusCode in clause 11.3 shall be included in the *ResponseConfirm* primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the *requestingEntity* based on *accessRightID*".

4) "Check syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.3.2.8.5 Update announced Resource

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" (clause 10.3.2.13) and then "Send ResponseConfirm primitive" (clause 10.3.2.17). The corresponding *statusCode* as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the *requestingEntity* based on *accessRightID*".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for update".

6) "Update the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.3.2.8.6 Delete announced Resource

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" (clause 10.3.2.13) and then "Send ResponseConfirm primitive" (clause 10.3.2.17). The corresponding *statusCode* as indicated in enumeration StatusCode in clause 11.3 shall be included in the *ResponseConfirm* primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of *requestingEntity* based on *accessRightID*".

4) "Check the syntax of the received message".

5) "Delete the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.3.2.9 DeAnnounce resource

Deannouncing a resource corresponds to removing the announcement from all the SCLs where the resource is announced. The procedure for deannouncing is the same as the procedure for updating an announcement, where the *sclList* element is set to the empty list.

10.3.2.10 Create the resource

A new resource shall be created and added to the addressed and existing parent resource. This shall modify the resource representation of the addressed parent resource, specifically, if the parent resource has a *lastModifiedTime* attribute this shall be set to same value as the *creationTime* attribute of the created resource. The following rules shall be applied:

If the created resource type has an id attribute (see Resource Attributes table), then the URI of the created resource shall be the URI of its parent resource with the URI-encoded id appended. (e.g.
http://sclbase.operator.org/applications/myAppID, for an application resource with id "myAppID" created in the parent resource http://sclbase.operator.org/applications).
If a resource with the same ID already exists in the addressed collection, the hosting SCL shall provide a new *id* that is unique within the collection.

- If the created resource type does not have an id-attribute, its URI shall be the URI of its parent with the type of the resource appended.

- If expirationTime attribute is present in the resource representation of the to be created resource and the expirationTime is set to a non-negative time, then an expiration timer shall be started by the hosting SCL. At timer expiration the related resource is deleted by "Delete the addressed resource".

For setting the attributes in the resource representation the following rules shall apply in CREATE request primitives:

M attribute

If the provided value is acceptable, the server shall use the provided value in the resource representation of the created resource.

If the provided value is not acceptable, and the response column is marked M* then the hosting SCL shall create the resource with a value for the attribute that is as close to the requested value as possible, as specified in clause 11.5 and the complete resource representation shall be returned in the confirm response.

O attribute

If a value is provided and accepted, then the server shall use the provided value in the resource representation of the created resource.

If a value is provided and this provided value is not acceptable, and the response column is marked M* then the hosting SCL shall create the resource with a value for the attribute that is as close to the requested value as possible, as specified in clause 11.5 and the complete resource representation shall be returned in the confirm response.

If the attribute is not present in the resource representation in the CREATE request and the response column is marked M or M* then the hosting SCL shall create the resource with the default value as indicated in clause 11.5. The hosting SCL does not have to return the complete resource representation in this case.

If the attribute is not present in the resource representation in the CREATE request and the response column is marked O then the hosting SCL shall create the resource without the attribute.

NP attribute

If the attribute is not present in the resource representation in the CREATE request, and the response column is marked M, then the hosting SCL shall create the resource with the default value as indicated in clause 11.5.

In particular, for attributes that contain reference to child/sub-resources, this involves creation of these sub-resources. A sub-resource shall be created with all the attributes set to default values, as specified in table 11.36, with the following additional constraints:

- Any sub-resource that has an *accessRightID* shall be created with the same value for the accessRightID as its parent.

- Any sub-resource with a *lastModifiedTime* and creationTime shall get the same value for these attributes as the corresponding attributes on the parent resource.

10.3.2.11 Create a collection resource representation

The SCL shall construct a resource representation of the collection resource as follows:

For each child resource the collection resource representation shall include the URI identifying the child resource.

However, the URI is only included if the *requestingEntity* has discovery permission on the child resource, i.e. the requestingEntity shall be one of the *permissionHolders* for a permission element of the permissions attribute, whose *permissionFlag* is not empty.

10.3.2.12 Create a successful ResponseConfirm

The receiver shall create a successful *ResponseConfirm* primitive with a *statusCode* indicating "STATUS_OK". The response shall include the representation of the addressed resource, if the hosting SCL modified any of the provided attributes as provided in the *RequestIndication*. The *ResponseConfirm* shall also include the URI of a created resource.

10.3.2.13 Create an unsuccessful ResponseConfirm

The receiver shall create an unsuccessful ResponseConfirm primitive with a statusCode indicating the detected error condition.

10.3.2.14 Read the addressed resource

An existing and accessible resource is addressed. The content of its attributes and a references to its sub-resources shall be read in form of resource representation.

10.3.2.15 Update the addressed resource

An existing and accessible resource is addressed. The content of its attributes shall be updated with the following rules in UPDATE request primitives:

M attribute

If the provided value is accepted, the server shall use the provided value in the resource representation of the updated resource.

If the provided value is not acceptable, and the response column is marked M* then the hosting SCL shall update the resource with a value for the attribute that is as close to the requested value as possible, as specified in clause 11.5 and the complete resource representation shall be returned in the confirm response.

O attribute

If a value is provided and the value is accepted, the server shall use the provided value in the resource representation of the updated resource.

If a value is provided and the provided value is not acceptable, and the response column is marked M* then the hosting SCL shall update the resource with a value for the attribute that is as close to the requested value as possible, as specified in clause 11.5 and the complete resource representation shall be returned in the confirm response.

If the attribute is not present in the resource representation in the UPDATE request and the response column is marked M or M* then the hosting SCL shall update the resource with the default value as indicated in clause 11.5. The hosting SCL does not have to return the complete resource representation in this case.

If the attribute is not present in the resource representation in the UPDATE request and the response column is marked O then the hosting SCL shall remove the attribute in the updated resource.

NP attribute

If the attribute is not present in the resource representation in the UPDATE request and the response column is marked M then the hosting SCL shall not update the attribute value. There is only one exception to this rule and that is the *lastModifiedTime* attribute. The hosting SCL shall set the *lastModifiedTime* to the current time whenever an update primitive is received.

If the attribute is present in the resource representation in the UPDATE request the presented value shall be ignored, i.e. the hosting SCL shall never update its resource representation based on the presence of an NP attribute value in an update.

General

If the *expirationTime* attribute is present and modified by the procedure and it is set to a non-negative time, then an expiration timer shall be re-started by the hosting SCL. At timer expiration the related resource is deleted by "Delete the addressed resource".

If the *announceTo* attribute is added or modified by the procedure, then the procedure as specified in "update of announce on request of application" is executed. If the *announceTo* attribute is removed by the procedure, the procedure as specified in "update announce on request of local SCL" is executed.

Post-Result Actions

If the *accessRightID* attribute, the *searchStrings* attribute or the *expirationTime* attribute of a resource is modified in the above actions and if the resource is being announced, then the hosting SCL shall update the corresponding announced resource by sending an UPDATE request.

10.3.2.16 Delete the addressed resource

An existing and accessible resource is addressed. The resource with all its attributes shall be deleted. Any expiration timer shall be stopped. This same procedure shall then be invoked (recursively) for each sub-resource of the deleted resource.

The parent resource of the addressed resource shall be updated to remove the reference to the deleted resource. If the parent resource has a *lastModificationTime* attribute then this attribute shall be set to the time of the deletion. The modification of the parent resource may trigger notifications. Refer to clause 10.25.7 for details.

If the resource is announced the SCL shall try to deannounce the resource as described in "DeAnnounce resource". Note that if it is not possible to deannounce the resource (e.g. since the SCL where the resource is announced cannot be reached), then the expiration timer of the announced resource will ensure that the announcement resource is removed.

Deletion of a resource may trigger notifications to be sent. Refer to clause 10.25.7.3 for details.

10.3.2.17 Send ResponseConfirm primitive

A ResponseConfirm primitive shall be sent back to the issuer.

10.3.2.18 Identify the managed remote entity and the management protocol

The hosting SCL shall identify the remote entity to be managed via the <*scl*> resource in the URI path as provided in the *targetID* primitive attribute. Then the hosting SCL shall determine the management protocol to be used for communicating with the remote entity based on the *mgmtProtocolType* attribute of the <scl> resource. If the remote entity cannot be identified, the hosting SCL shall reject the request with the *statusCode* set to "STATUS_NOT_FOUND" in the ResponseConfirm primitive.

10.3.2.19 Locate the MO information to be managed on the remote entity

The hosting SCL shall locate the MO information to be managed on the remote entity by the *originalMO* attribute of the <*mgmtObj*> or <*parameters*> resource addressed by the URI provided in the *targetID* primitive attribute. In the case that the *targetID* addresses an <*moAttribute*> or a <*parameters*> resource without *originalMO* attribute, the hosting SCL shall locate the MO information on the remote entity through the *originalMO* attribute of the closest ancestor <*mgmtObj*> (or <*parameters*>) resource of the addressed <*moAttribute*> or <*parameters*>, combining with their relative position in the MO tree. If the MO information cannot be located, the hosting SCL shall reject the request with the *statusCode* set to "STATUS_NOT_FOUND" in the ResponseConfirm primitive.

10.3.2.20 Establish a management session with the remote entity

If there is no existing management session between the hosting SCL and the remote entity to be managed, the hosting SCL shall also trigger the remote entity to establish a management session with the hosting SCL by sending triggering message to the address provided in the *remTriggerAddr* attribute of the parent resource <*scl*> using the determined management protocol (e.g. "DM Notification" [67], "ACS Connection Initiation" [49]). If the management session cannot be established within a limited time span as per local policy, the hosting SCL shall reject the request with the *statusCode* set to "STATUS_GATEWAY_TIMEOUT" in the ResponseConfirm primitive.

10.3.2.21 Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive

The hosting SCL shall send the management request(s) to the remote entity in the established management session over mId reference point in order to perform the management operation as requested by the received *RequestIndication* primitive. The management request shall address the MO information on the remote entity as determined in clause 10.3.2.18 or in the primitive specific clauses. The management request being used is specific to the underlying management protocol according to a pre-defined mapping relationship with the *RequestIndication* primitive as described in table 10.3. The internal data structure of the MO addressed by the management request shall be determined based on the mapping relationship as specified in annex E for the ETSI specific <*mgmtObj*> resource instances or based on the generic mapping rule as specified in TS 102 690 [2], clause 8.2.3.26, for any other <*mgmtObj*> resource instances. The hosting SCL shall extract the management results received from the remote entity in order to prepare a *ResponseConfirm* primitive to be sent to the issuer later. Unless explicitly stated, if the management request cannot be performed successfully, the hosting SCL shall reject the *RequestIndication* primitive with the proper *statusCode* in the *ResponseConfirm* primitive according to the mapping relationship with underlying management protocol as defined in tables 10.4 and 10.5.

Table 10.3: Generic mapping relationship between request indication primitives and underlying protocol specific management commands or methods

RequestIndication primitive	OMA-DM command [68]	BBF TR-069 RPC method [49]
MgmtObjCreateRequestIndication	Add	AddObject
MgmtObjRetrieveRequestIndication	Get	GetParameterValues GetParameterAttributes (see note 1)
MgmtObjUpdateRequestIndication	Replace	SetParameterValues SetParameterAttributes (see note 2)
MgmtObjDeleteRequestIndication	Delete	DeleteObject
MgmtObjExecuteRequestIndication	Exec	n/a (see note 3)
NOTE 1: Depending on the nature (value or attribute) of the MO information to be retrieved from the remote entity, the hosting SCL may choose either or both of the two methods to perform the management operation. NOTE 2: Depending on the nature (value or attribute) of the MO information to be updated on the remote entity, the hosting SCL may choose either or both of the two methods to perform the management operation. NOTE 3: Execution of BBF TR-069 RPCs shall be mapped to <mgmtCmd> primitives as described in clause 9.28. NOTE 4: For ETSI specific <mgmtObj> resource instances, a RequestIndication primitive may be mapped to one or multiple protocol specific management commands or methods as described in annex E.		

Table 10.4: ETSI statusCode mapping relationship with OMA-DM statusCode

Status Code defined for OMA-DM commands [68]	Status Code in ResponseConfirm primitive
(200) OK	STATUS_OK
(202) Accepted for processing	STATUS_ACCEPTED
(213) Chunked item accepted	(not used)
(215) Not executed	STATUS_INTERNAL_SERVER_ERROR
(216) Atomic roll back OK	STATUS_INTERNAL_SERVER_ERROR
(217) OK with inherited ACL	STATUS_OK
(401) Unauthorized	STATUS_PERMISSION_DENIED
(403) Forbidden	STATUS_FORBIDDEN
(404) Not Found	STATUS_NOT_FOUND
(405) Command not allowed	STATUS_METHOD_NOT_ALLOWED
(406) Optional feature not supported	STATUS_BAD_REQUEST
(407) Authentication required	STATUS_PERMISSION_DENIED
(413) Request entity too large	STATUS_BAD_REQUEST
(414) URI too long	STATUS_BAD_REQUEST
(415) Unsupported media type or format	STATUS_UNSUPPORTED_MEDIA_TYPE
(418) Already exists	STATUS_OK (if the existing MO is the same as the one to be added) STATUS_CONFLICT (if the existing MO is different than the one to be added)
(420) Device full	STATUS_SERVICE_UNAVAILABLE
(424) Size mismatch	STATUS_INTERNAL_SERVER_ERROR
(425) Permission denied	STATUS_UNAUTHORIZED
(500) Command failed	STATUS_INTERNAL_SERVER_ERROR
(510) Data store failure	STATUS_INTERNAL_SERVER_ERROR
(516) Atomic roll back failed	STATUS_INTERNAL_SERVER_ERROR

Table 10.5: ETSI statusCode mapping relationship with BBF TR-069 statusCode

Status Code defined for BBF TR-069 methods [49]	Status Code in ResponseConfirm primitive
(9001) Request denied (no reason specified)	STATUS_FORBIDDEN
(9002) Internal error	STATUS_INTERNAL_SERVER_ERROR
(9003) Invalid arguments	STATUS_BAD_REQUEST
(9004) Resources exceeded	STATUS_SERVICE_UNAVAILABLE
(9005) Invalid parameter name	STATUS_BAD_REQUEST
(9006) Invalid parameter type (associated with SetParameterValues)	STATUS_BAD_REQUEST
(9007) Invalid parameter value (associated with SetParameterValues)	STATUS_BAD_REQUEST
(9008) Attempt to set a non-writable parameter (associated with SetParameterValues)	STATUS_BAD_REQUEST
(9009) Notification request rejected (associated with SetParameterAttributes method)	STATUS_FORBIDDEN

10.3.2.22 Identify the managed remote entity and the management protocol

After the hosting SCL (i.e. NSCL) establishes a management session with the remote entity, it shall enter into this step to convert the received mgmtCmd-related primitives to corresponding BBF TR-069 [49] RPC and shall send the converted RPC to the remote entity via the established management session. Management command conversion and execution shall be required for the following primitives:

- **mgmtCmdExecute:** Each of following BBF TR-069 [49] RPCs (FactoryReset, Reboot, Upload, Download, ScheduleDownload, ScheduleInform, and ChangeDUState [49]) has a corresponding <*mgmtCmd*> resource. The hosting SCL shall convert the received mgmtCmdExecute primitive into corresponding BBF TR-069 [49] RPC and creates a local <*execInstance*> resource for the converted RPC. The hosting SCL shall assign and maintain a CommandKey for each converted RPC and corresponding <*execInstance*>.

- **mgmtCmdDelete:** BBF TR-069 [49] only defines three cancellable commands (Upload, Download, and ScheduleDownload). As a result, only the "*cmdType*" attribute of the <*mgmtCmd*> resource as requested to be delete represents Upload, Download, or ScheduleDownload, this step shall be required for the hosting SCL to cancel the corresponding command at the remote entity by converting the received mgmtCmdDelete primitive to BBF TR-069 CancelTransfer RPC. If such a <*mgmtCmd*> resource to be deleted has multiple unfinished <*execInstance*>, multiple CancelTransfer RPC will be generated and each CancelTransfer RPC corresponds to a single cancellable <*execInstance*> resource. According to BBF TR-069 [49], the CommandKey which the hosting SCL maintains for each <*execInstance*> resource shall be contained in the CancelTransfer to instruct the remote entity which command is to be cancelled.

- **execInstanceDelete:** This primitive requests to delete an <*execInstance*> resource. If the <*execInstance*> represents one of the three cancellable RPCs as defined in BBF TR-069 [49] and is not complete, the hosting SCL shall generate a CancelTransfer RPC in order to delete the unfinished RPC on the remote entity as represented by the <*execInstance*> resource.

- **execInstanceExecute:** This primitive requests to explicitly cancel an unfinished RPC on the remote entity. This primitive is only valid for <*execInstance*>, which corresponds to BBF TR-069 [49] Upload, Download, or ScheduleDownload RPC.

The status of each <execInstance> shall be the following (i.e. the value of "*execStatus*" attribute of an <*execInstance*> resource). The transition diagram is illustrated in figure 10.1:

- **Initiated:** It means the <*execInstance*> is just initiated on the remote entity by the hosting SCL, but it is not started yet. It shall enter to this state after the hosting SCL receives mgmtCmdExecute primitive from an M2M Network Application and processes *this primit*ive successfully. This state may move to "Started", "Cancelling" or end:

    - "Initiated" -> "Started": It happens when the remote entity starts an previously initiated command:

        - If there is no delay specified in the command sent to the remote entity, "Initiated" shall transfer to "Started" immediately.

        - If there is a delay specified in the command sent to the remote entity, "Initiated" shall transfer to "Started" after such an amount of delay. There is such a delay parameter in three BBF TR-069 commands: Download, Upload, and ScheduleDownload.

    - "Initiated"-> "Cancelling": It happens when the hosting SCL issues BBF TR-069 CancelTransfer RPC to the remote entity, triggered by receiving mgmtCmdDelete, execInstanceDelete, or execInstanceExecute primitive from an M2M Network Application.

    - "Initiated" -> End: It happened when the <*execInstance*> resource is removed due to reasons:

        - The expirationTime of the resource <*execInstance*> or the expirationTime of its parent resource is passed.

        - The hosting SCL receives mgmtCmdDelete or execInstanceDelete primitive from an M2M Network Application and <*execInstance*> represents a non-cancellable BBF TR-069 [49] RPC.

- Started: It means the <*execInstance*> is started at the remote entity. This state may move to "Finished", "Cancelling", or End:

    - "Started" ->"Finished": The corresponding RPC of an <*execInstance*> resource completes on the remote entity and the hosting SCL receives a response from the remote entity.

    - "Started" ->"Cancelling": It happens when the hosting SCL issues BBF TR-069 CancelTransfer RPC to the remote entity, triggered by receiving mgmtCmdDelete, execInstanceDelete, or execInstanceExecute primitive from an M2M Network Application:

        - According to BBF TR-069 [49], a transfer command (Upload, Download, or ScheduleDownload) after it is started, can be cancelled. But in some cases, it may not be allowed to cancel, for example, an uninterruptible active transfer.

  - "Started"->End: after the command is started, the corresponding <*execInstance*> resource shall be deleted due to reasons such as:

    - the expirationTime of the resource <*execInstance*> or the expirationTime of its parent resource is passed; or

    - the hosting SCL receives mgmtCmdDelete or execInstanceDelete primitive from an M2M Network Application and <*execInstance*> represents a non-cancellable BBF TR-069 [49] RPC.

- Cancelling: It enters to this state from "Initiated" or "Started" when the hosting SCL receives mgmtCmdDelete, execInstanceDelete, or execInstanceExecute primitive and accordingly conducts command cancellation by issuing BBF TR-069 CancelTransfer to the remote entity if the <*execInstance*> resource represents a cancellable resource. After the cancellation is complete, the <*execInstance*> resource shall be deleted no matter the cancellation is successful or not.

- Finished: It means that the BBF TR-069 [49] RPC corresponding to an <*execInstance*> is finished at the remote entity. This state can go to end because the corresponding <*execInstance*> resource is deleted due to:

  - the expirationTime of the resource <*execInstance*> or the expirationTime of its parent resource is passed; or

  - the hosting SCL receives mgmtCmdDelete or execInstanceDelete primitive from an M2M Network Application.

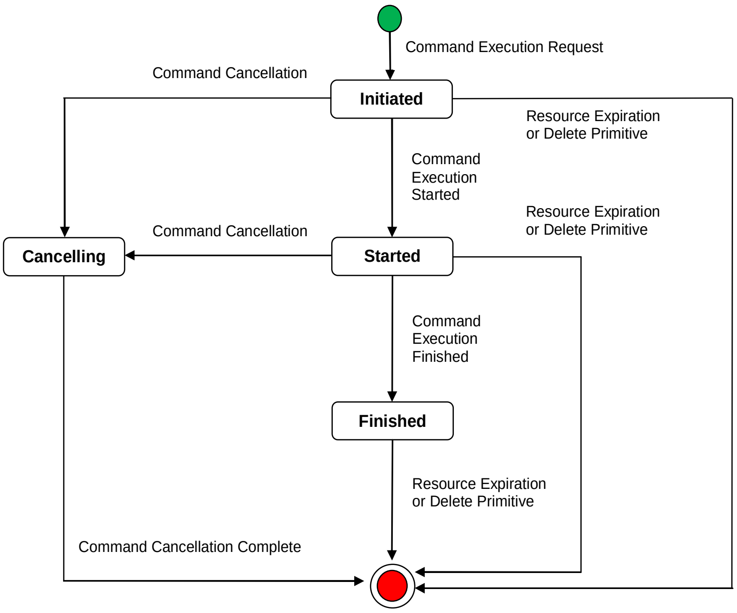

Command Execution Request
Command Cancellation
Initiated
Resource Expiration or Delete Primitive
Command Execution Started
Resource Expiration or Delete Primitive
Command Cancellation
Cancelling
Started
Command Execution Finished
Finished
Resource Expiration or Delete Primitive
Command Cancellation Complete

Figure 10.1: State Transition Diagram of a Management Command Represented by an <execInstance> Resource

10.3.2.23 SCL retargeting to an application

The SCL shall check if the following conditions apply:

- the targetID does not identify a resource that is stored in the local SCL; and

- the targetID in the received request is hierarchically subordinate to the <application> resource; and

- the targetID does not correspond to an attribute in the <application> resource according to the rules defined in [partial addressing].

If these conditions apply then the SCL shall try to retarget the request to the application by executing the following steps in order:

1) The hosting SCL shall check if the <application> resource in the targetID contains an aPoc attribute with a non-empty value. If not, then the request is rejected with a STATUS_NOT_FOUND.

2) The hosting SCL shall check if the <application> resource in the targetID contains an aPoCPaths attribute. If not then the hosting SCL reject the request with a STATUS_NOT_FOUND.

3) The hosting SCL shall sort the aPoCPath elements, in the aPoCPaths attribute, by decreasing length of their path element and use this ordered list in the next step.

4) For each aPocPath element in the list derived in step 3, the hosting SCL shall check if the aPocPath matches the targetID. The matching algorithm shall depend on the value of the aPocHandling attribute in the <sclBase> resource of the hosting SCL:

   a) If the *aPocHandling* attribute has the value "DEEP", then the aPocPath shall be considered a match if the concatenation of the resource URI of the *<application>* resource and the *path* in the *aPocPath* element is a prefix of the received *targetID* primitive attribute.

   b) If the *aPocHandling* attribute has the value SHALLOW or if the sclBase resource does not have an *aPocHandling* attribute, then the aPocPath shall be considered a match in the following cases:

      i) in case the *path* element does not end in a slash ("/"); if the concatenation of the resource URI of the *<application>* resource and the *path* in the *aPocPath* element is an exactly match for the received *targetID* primitive attribute; or

      ii) in case the path element does end in a slash; if the concatenation of the resource URI of the <application> resource and the *path* in the *aPocPath* element is a prefix match of received *targetID* primitive attribute and there is at most one URI path element in the targetID after the matched part.

   c) If there is a match then the hosting SCL shall check if the *aPocPath* element ,whose *path* element matched, has a *accessRightID* element:

      i) If it does not have an *accessRightID*, then continue from step 6.

      ii) If it does have an *accessRightID* then the hosting SCL shall apply the common procedure "Check authorization of the *requestingEntity* based on the *accessRightID*". However, in this case the *accessRightID* in question shall be the *accessRightID* from the matching *aPocPath* element. If access is granted then continue from step 7.

5) If there was no match in step 4, then the request shall be rejected with a STATUS_NOT_FOUND.

6) **Apply Application AR:** "check authorization of the *requestingEntity* based on the *accessRightID*", where the *accessRightID* is the *accessRightID* of the *<application>* resource.

7) **Request authorized:** The hosting SCL shall retarget the request, i.e. in the targetID it shall substitute the resource URI of the registered application path prefix (as registered by the application to its SCL e.g. /http://gsc1SCL1/applications/dev_app_1/) with the URI specified in the *aPoC* attribute of the *<application>* resource. Then the hosting SCL shall route the request based on the resulting URI. On the dIa, the routing may be based on internal local area network address schemes. On the mIa, the resulting address shall be routed according to normal URI resolving rules (e.g. applying DNS lookup, etc.).

8) The hosting SCL shall forward the request to the address determined in the previous step, using the retargeted URI as the requestURI, using the same REST method as received on the incoming request and using the same payload.

9) The hosting SCL shall wait for the REST result from retargeted request.

10) The hosting SCL shall map the REST response to a response confirm primitive. Any payload in the REST response shall be carried as a payload in the ResponseConfirm.

11) The hosting SCL shall forward the ResponseConfirm primitive to the issuer of the primitive request.

10.3.3 Receiver SCL actions

10.3.3.1 Re-targeting

If the *targetID* in the request does not start with the sclBase URI of the receiver , the receiver SCL shall forward the request or shall serve the request locally (see below).

If the *targetID* in the request starts with the sclBase URI of the receiver, then the receiver SCL shall handle the request locally.

Acting as an issuer the SCL shall perform the following procedures:

1) "Send a RequestIndication to the receiver SCL".

2) "wait for ResponseConfirm primitive".

When the ResponseConfirm is received the receiver SCL shall:

1) primitive specific procedure: Forward the ResponseConfirm to the original issuer SCL.

The receiver SCL may serve a RETRIEVE request locally. If the addressed resource in the targetID is a resource that is cached in the receiver SCL, the receiver SCL may handle the RETRIEVE request indication locally, without forwarding the request to the hosting SCL.

DELETE, UPDATE, EXEC and CREATE requests shall be forwarded, however, in these cases the receiver SCL may perform some checks normally performed in the hosting SCL, if it has the information to perform these checks, such checks may lead to rejecting the request with a unsuccessful ResponseConfirm without forwarding the request.

For example the receiver SCL may:

- Check the syntax of received message.
- Check validity of resource representation.
- Check authorization of requestingEntity based on accessRightID.
- Check authorization of requestingEntity based on selfPermission.

The latter two actions require the receiving SCL to have a partial (accessRightID / selfPermissions) or full cached and up-to-date copy of the addressed resource locally available.

10.4 <sclBase> resource and management procedures

10.4.1 <sclBase> resource

The *sclBase* resource shall contain the following sub-resources and attributes.

Table 10.6: Attributes and resource for the *sclBase*

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
sclsReference	N/A	NP	M	Reference to sub-resource "scls" in table 11.37
applicationsReference	N/A	NP	M	Reference to sub-resource "applications" in table 11.37
containersReference	N/A	NP	M	Reference to sub-resource "containers" in table 11.37
groupsReference	N/A	NP	M	Reference to sub-resource "groups" in table 11.37
accessRightsReference	N/A	NP	M	Reference to sub-resource "accessRights" in table 11.37
subscriptionsReference	N/A	NP	M	Reference to sub-resource "subscriptions" in table 11.37
discoveryReference	N/A	NP	M	Reference to "discovery" resource in table 11.37
accessRightID	N/A	O	O	See table 11.36
searchStrings	N/A	O	M	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36
aPocHandling	N/A	O	O	See table 11.36

10.4.2 sclBaseCreate

The <*sclBase*> resource shall not be created via the API.

Instead the <*sclBase*> resource is created by out-of-band means (e.g. a service provider may have administrative commands to create an sclBase resource or the sclBase is created when the device or gateway is first started) . When the sclBase is created, all its child resources (scls, applications, containers, groups, accessRights, subscriptions, discovery) shall be created as well and their references shall be set in the <sclBase> resource.

If the NSCL wants to be discoverable using searchStrings in the G/DSCL that are registered to the NSCL, then the NSCL shall include a *searchStrings* attribute in the *sclBase* resource.

The *searchStrings* attribute as specified in the D/GSCL's *sclBase* resource is never used for discovery purposes. If the D/GSCL wants to be discoverable using searchStrings in the NSCL, then the D/GSCL shall provide a *searchStrings* attribute in the sclCreateRequestIndication, i.e. at SCL registration.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.4.3 sclBaseRetrieve

10.4.3.1 sclBaseRetrieveRequestIndication

This primitive is used to retrieve an <*sclBase*> Resource. The SCL primitive shall comply with the following.

Table 10.7: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.8: Primitive sclBaseRetrieveRequestIndication

SCL Primitive: sclBaseRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Indicates the sclBase resource to be retrieved
primitiveType	M	CLBASE_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: If the <sclBase> resource resides on the GSCL or DSCL and the requesting entity represents the NSCL with which the hosting SCL is registered, then the request shall be allowed, i.e. step 4 shall be skipped and the sequence continues with step 5.

4) "Check authorization of the requestingEntity based on accessRightID".

5) "Check the syntax of received message".

6) "Read the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.4.3.2 sclBaseRetrieveResponseConfirm (successful case)

Confirms the retrieval of an *sclBase* Resource. The SCL primitive shall comply with the following.

Table 10.9: sclBaseRetrieveResponseConfirm, successful case

SCL primitive: sclBaseRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	Indicates the type of primitive: SCLBASE_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
sclBase	M	The sclBase resource representation as indicated in clause 10.4.1

10.4.3.3 sclBaseRetrieveResponseConfirm (unsuccessful case)

This response is triggered by an error in the sclBaseRetrieveRequestIndication procedure. The SCL primitive shall comply with table 10.10.

Table 10.10: sclBaseRetrieveResponseConfirm, unsuccessful case

SCL primitive: sclBaseRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCLBASE_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.4.4 sclBaseUpdate

10.4.4.1 sclBaseUpdateRequestIndication

This clause describes the full update of an *sclBase* resource.

The SCL primitive shall comply with the following.

Table 10.11: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.12: Primitive sclBaseUpdateRequestIndication

SCL Primitive: sclBaseUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Indicates the sclBase resource to be updated
primitiveType	M	SCLBASE_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
sclBase	M	The sclBase resource representation as indicated in clause 10.4.1

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource"
3) Primitive specific operation: If the <*sclBase*> resource resides on the GSCL or DSCL and the requesting entity represents the NSCL with which the hosting SCL is registered, then the request shall be allowed, i.e. step 4 shall be skipped and the sequence continues with step 5.
4) "Check authorization of the requestingEntity based on accessRightID".
5) "Check the syntax of received message".
6) "Check validity of the resource representation for UPDATE".
7) "Update the addressed resource". Primitive specific operations: if the requestingEntity is not the NSCL with which the hosting SCL is registered, then the *aPocHandling* attribute shall not be modified by this operation.
8) "Create a successful response Confirm".
9) "Send response Confirm primitive".

10.4.4.2 sclBaseUpdateResponseConfirm (successful case)

Confirms the update of an sclBase Resource. The SCL primitive shall comply with the following.

Table 10.13: sclBaseUpdateResponseConfirm, successful case

SCL primitive: sclBaseUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCLBASE_UDPATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
sclBase	O	The sclBase resource representation as indicated in clause 10.4.1. This shall present if any of the attributes provided in the resource representation in the sclBaseUpdateRequestIndication was modified by the hosting SCL

10.4.4.3 sclBaseUpdateResponseConfirm (unsuccessful case)

This response is triggered by the sclBaseRetrieveRequestIndication. The SCL primitive shall comply with table 10.14.

Table 10.14: sclBaseUpdateResponseConfirm, unsuccessful case

SCL primitive: sclBaseUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCLBASE_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.4.5 sclBaseDelete

The <sclBase> resource shall not be deleted via the API.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

Instead the sclBase may be deleted via some out-of-band means. When the sclBase is deleted, all its descendant resources shall be deleted as well, and the actions as described in 'delete addressed resource' are applied for each of these resources. However, the hosting SCL may optimize performance by skipping any actions that have only local effects (for example, if a resource is deleted which is being subscribed by another resource in the same sclBase and by a resource in a remote SclBase, a notify only has to be sent to the remote resource).

10.5 scls resource and management procedures

10.5.1 scls resource

The scls resource shall contain the following sub-resources and attributes.

Table 10.15: Attributes and primitive for scls

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
sclCollection	N/A	NP	M	References to sub-resources "<scl> in table 11.37
subscriptionsReference	N/A	NP	M	Reference to sub-resource subscriptions in table 11.37
mgmtObjReference	N/A	NP	O	Reference to sub-resource mgmtObjs in table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.5.2 sclsCreate

The scls collection resource shall not be created directly via the API. It is created whenever the parent sclBase resource is created. The attributes are set to their default values as specified in table 10.15. The accessRightID is initialized to same value as the accessRightID in the parent.

Note that since the sclBase parent is not created explicitly via the API, the creator of the sclBase may also initialize the scls resource with an accessRightID, e.g. according to internal configuration.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.5.3 sclsRetrieve

10.5.3.1 sclsRetrieveRequestIndication

This primitive is used to retrieve an scls collection Resource. The SCL primitive shall comply with table 10.16.

Table 10.16: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.17: sclsRetrieveRequestIndication

SCL primitive: sclsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Indicates the scls collection resource to be retrieved
primitiveType	M	SCLS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive to the Receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The **receiver** shall not execute following steps if the request is re-targeted:

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Create a collection resource representation".

6) "Create a successful ResponseConfirm" with the created collection resource representation.

7) "Send response Confirm primitive".

10.5.3.2 sclsRetrieveResponseConfirm (successful case)

Confirms the retrieval of an sclBase Resource. The SCL primitive shall comply with table 10.18.

Table 10.18: sclsRetrieveResponseConfirm, successful case

SCL primitive: sclsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCLS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
scls	M	The scls resource representation as indicated in clause 10.5.1

10.5.3.3 sclsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the sclsRetrieveRequestIndication. The SCL primitive shall comply with table 10.19.

Table 10.19: sclsRetrieveResponseConfirm, unsuccessful case

SCL primitive: sclsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCLS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.5.4 sclsUpdate

10.5.4.1 sclsUpdateRequestIndication

It updates the provided attributes of an existing collection resource in a Service Capability Layer. This primitive describes the full update. In addition to a full update, it is also possible to update a single attribute or part of an attribute.

See clause 10.39.5 for details.

The SCL primitive shall comply with table 10.20.

Table 10.20: Applicability of primitive

SCL Primitive: sclsUpdateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.21: sclsUpdateRequestIndication

SCL Primitive: sclsUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Link to the scls collection resource to be updated
primitiveType	M	Indicates the SCLS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
scls	M	An scls resource representation as defined in clause 10.5.1 that replaces the current representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive to the Receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted:

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Validate resource representation for update".

6) "Update the addressed resource".

7) "Create successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

Note that since the resource representation only contains one Read-Write attribute (accessRightID), this shall be the only attribute that can be updated using this procedure.

10.5.4.2 sclsUpdateResponseConfirm (successful case)

Successful response on updating of an existing Collection resource in a Service Capability Layer.

Table 10.22: sclsUpdateResponseConfirm, successful case

SCL Primitive: sclsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	Indicates the SCLS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
scls	O	An scls resource full representation as defined in clause 10.5.1. This shall be present if the hosting SCL modified any of the attributes provided by the Issuer

10.5.4.3 sclsUpdateResponseConfirm (unsuccessful case)

Unsuccessful response after an attempt to update an Collection resource in a Service Capability Layer.

Table 10.23: sclsUpdateResponseConfirm, unsuccessful case

SCL Primitive: sclsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCSLS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.5.5 sclsDelete

The scl collection resource shall not be deleted via the API.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.6 <scl> resource and management procedures

10.6.1 <scl> resource

The structure of the <scl> resource depends on whether the resource is hosted by an NSCL or a D/GSCL.

When hosted in an NSCL the <scl> resource shall contain the following sub-resources and attributes.

Table 10.24: Resource description for *<scl> (NSCL)*

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
containersReference	NP	NP	M	Reference to sub-resource containers in table 11.37
groupsReference	NP	NP	M	Reference to sub-resource groups in table 11.37
applicationsReference	NP	NP	M	Reference to sub-resource application in table 11.37
accessRightsReference	NP	NP	M	Reference to sub-resource accessRights in table 11.37
subscriptionsReference	NP	NP	M	Reference to sub-resource subscriptions in table 11.37
mgmtObjsReference	NP	NP	M	Reference to sub-resource mgmtObjs in table 11.37
notificationChannelsReference	NP	NP	M	Reference to sub-resource notificationChannels in table 11.37
m2mPocsReference	NP	NP	M	Reference to sub-resource m2mPocs in table 11.37
attachedDevicesReference	NP	NP	M	Reference to sub-resource attachedDevices in table 11.37
sclId	M	NP	M	The SCL-ID of the issuer SCL. A globally unique id shall be provided by the issuer in the CREATE request.
pocs	O	O	M	See table 11.36
remTriggerAddr	O	O	O	See table 11.36
onlineStatus	NP	NP	M	See table 11.36
serverCapability	NP	NP	M	See table 11.36
link	M	NP	M	See table 11.36. The issuer shall provide a unique link URI. It shall be unique in the sense that there no other SCL that either uses the same URI or uses a URI that is a prefix of this URI.
schedule	O	O	O	See table 11.36
expirationTime	O	O	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
locTargetDevice	O	O	O	See table 11.36
mgmtProtocolType	M	M	M	See table 11.36
integrityValResults	O	O	O	See table 11.36
aPocHandling	NP	NP	O	See table 11.36

When hosted by a D- or GSCL, the <scl> resource shall contain the following sub-resources and attributes.

Table 10.24a: Resource description for *<scl> (D/GSCL)*

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
containersReference	N/A	N/A	M	Reference to sub-resource containers in table 11.37
groupsReference	N/A	N/A	M	Reference to sub-resource groups in table 11.37
applicationsReference	N/A	N/A	M	Reference to sub-resource application in table 11.37
accessRightsReference	N/A	N/A	M	Reference to sub-resource accessRights in table 11.37
subscriptionsReference	N/A	N/A	M	Reference to sub-resource subscriptions in table 11.37
mgmtObjsReference	N/A	N/A	NP	Reference to sub-resource mgmtObjs in table 11.37
notificationChannelsReference	N/A	N/A	M	Reference to sub-resource notificationChannels in table 11.37
m2mPocsReference	N/A	N/A	NP	Reference to sub-resource m2mPocs in table 11.37
attachedDevicesReference	N/A	N/A	NP	Reference to sub-resource attachedDevices in table 11.37
sclId	N/A	N/A	M	The value shall be the uri-encoded value of the sclBase URI of the NSCL with which this D/GSCL registerd
pocs	N/A	N/A	M	See table 11.36. The attribute value shall contain the empty collection.
remTriggerAddr	N/A	N/A	NP	See table 11.36
onlineStatus	N/A	N/A	M	See table 11.36. The attribute value shall be ONLINE.
serverCapability	N/A	N/A	M	See table 11.36. The attribute value shall be TRUE.
link	N/A	N/A	M	See table 11.36. The value shall be the sclBase URI of the NSCL with which this G/DSCL registered.
schedule	N/A	N/A	NP	See table 11.36
expirationTime	N/A	N/A	M	See table 11.36. This value should be the same as the expiration time of the registration resource
accessRightID	N/A	N/A	O	See table 11.36. This value should be the same value as the accessRightID of the sclBase resource with which the D/GSCL registered.
searchStrings	N/A	N/A	M	See table 11.36. This value should be the same value as the searchString of the sclBase resource with which the D/GSCL registered.
creationTime	N/A	N/A	M	See table 11.36
lastModifiedTime	N/A	N/A	M	See table 11.36
locTargetDevice	N/A	N/A	NP	See table 11.36
mgmtProtocolType	N/A	N/A	NP	See table 11.36.
integrityValResults	N/A	N/A	NP	See table 11.36
aPocHandling	N/A	N/A	NP	See table 11.36

10.6.2 sclCreate

10.6.2.1 sclCreateRequestIndication

This procedure is used for creating an SCL resource of the Issuer-SCL at the Hosting-SCL. The primitive shall comply with table 10.25. The Hosting SCL shall use default values for optional parameters that are not specified.

If the receiver supports integrity validation, the receiver shall have retrieved the security attributes of the Issuer from the MAS during the authentication procedure. The security attributes of the Issuer include IValCapability and an IVal Key.

Table 10.25: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL (D/GSCL)
Receiver	N/A	N/A	Hosting SCL (NSCL)

Table 10.26: sclCreateRequestIndication

SCL Primitive: sclCreateRequestIndication		
Primitive Attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the primitive
targetID	M	The URI of the target entity where the SCL resource shall be created This request shall address <sclBase>/scls/ of the target SCL
primitiveType	M	SCL_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
scl	M	The scl resource to be created under <sclBase>/scls/ in the Hosting SCL

The **issuer** shall execute the following steps in order. The issuer SCL shall have successfully established secure communication as defined in clause 8 with the receiver prior to the following operations:

1) "Compose RequestIndication primitive": Primitive specific operation:

   a) In the RequestIndication, the issuer shall provide its SCL-ID in the *sclId* attribute as the name of the new resource that will be created in the Hosting SCL. The issuer obtains the SCL-ID from M2M service bootstrapping or provisioning (clause 6).

   b) The issuer shall include its < *sclBase* > in the "link" attribute.

   c) The issuer shall fill the device management protocol that it supports in the "mgmtProtocolType" attribute. The value may be from a DM client, or provisioned.

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

Post Response operation:

1) Upon receiving a successful response, the Issuer SCL shall create a new local *scl* resource, using the *<sclBase>* of the Hosting SCL to identify the local resource.

2) The values of the attributes for the local *<scl>* resource shall be assigned default values initially. With the exception of the expirationTime which is set as follows:

   a) If the ResponseConfirm received in step 3 above contains a resource representation then the issuer SCL shall set the value of the expirationTime attribute of the local *scl* resource to the value as received present in the resource representation in the ResponseConfirm.

   b) If the ResponseConfirm received in step 3 did not contain a resource representation then the issuer SCL shall set the value of the expirationTime attribute of the local *scl* resource to the value as present in the resource representation in the sclCreateRequest that was send in step 1.

c) Otherwise the expirationTime attribute of the local *scl* shall be set to a default value. The value will later (in step 14) be overridden.

d) If expirationTime attribute in the local *scl* resource is set to a non-negative time, then an expiration timer shall be started by the issuer SCL. At timer expiration the related resource is deleted by "Delete the addressed resource".

3) The Issuer SCL shall set the value of *link*" for the local <*scl*> resource to the *sclBase* resource URI of the receiver SCL.

4) Note that unless the issuer SCL performs a re-registration before the current registration expires, the issuer SCL automatically becomes unregistered. The registration expiration time is obtained from the *expirationTime* attribute as received in the sclCreateResponseConfirm (if this contains a resource representation) or from the *expirationTime* attribute as sent in the sclCreateRequestIndication (if the sclCreateResponseConfirm does not include a resource representation). The re-registration follows the sclUpdate procedure as described below.

5) The issuer SCL shall perform a RETRIEVE request on the *sclBase* resource of the Hosting SCL (sclBaseRetrieveRequestIndication) to request the *sclBase* resource attribute values.

6) The Receiver SCL shall execute the steps as specified for a receiver in the sclBaseRetrieveRequestIndication. For example, it shall validate that the Issuer SCL has the access rights to read the attributes of its <*sclBase*> resource.

7) Upon receiving an unsuccessful sclBaseRetrieveResponseConfirm, the issuer SCL shall continue on step 10. The issuer SCL may retry the post-response operation at a later time, starting from step 5. The issuer SCL shall only do this if it has reasons to believe the error is of a transient nature. For example, on receiving a STATUS_SERVER_UNAVAILABLE.

8) Upon receiving the successful sclBaseRetrieveResponseConfirm, the Issuer SCL shall update the value of the attribute(s) of its local SCL resource to match the received values from the *sclBase* resource representation in the sclBaseRetrieveResponseConfirm.

9) Specifically, the Issuer SCL shall set the value of the *searchStrings* and *accessRightID* attributes in its local *scl* resource to the value of the *searchStrings* and *accessRightID* attributes in *sclBase* resource representation received in the sclBaseRetrieveResponseConfirm from step 8.

9a) The issuer SCL may subscribe to it's the *sclBase* resource corresponding to the Hosting SCL if it wants to be notified when its *sclBase* resource is updated (e.g. when it is the searchStrings attribute is changed). Upon such a notification the issuer SCL shall update its local *scl* resource to match the changed in the *sclBase* resource

10) The issuer SCL may subscribe to its own <*scl*> resource created in the Hosting SCL if it wants to be notified when its *scl* resource is removed (e.g. when it is unregistered on request of an application that is authorized to do so).

11) If the ResponseConfirm received in step 3 contains a resource representation and the that resource representation contains a *aPocHandling* attribute value, then the issuer SCL shall set the *aPocHandling* attribute in its local <*sclBase*> resource to the received value.

12) If the ResponseConfirm received in step 3 contains a resource representation and the that resource representation does **not** contain a *aPocHandling* attribute value, then the issuer SCL shall set the *aPocHandling* attribute in its local <*sclBase*> resource to "SHALLOW".

13) If the ResponseConfirm received in step 3 did not contain a resource representation, then the issuer SCL shall perform the procedures as described in clause 10.6.3 with the targetID being the resourceURI received in step 3 of the sclCreate procedure:

a) If the sclRetrieve procedure completed successfully, the issuer SCL shall set the *aPocHandling* attribute in its local <*sclBase*> resource to the same value as the value of the *aPocHandling* attribute received in the resource representation of the <*scl*> resource received in the ResponseConfirm. Likewise, the issuer SCL shall set the *expirationTime* attribute in its local <*sclBase*> resource to the same value as the value of the *expirationTime* attribute received in the resource representation of the <*scl*> resource received in the ResponseConfirm.
If the *expirationTime* attribute in the local *scl* resource is set to a non-negative time, then an expiration timer shall be re-started by the issuer SCL. At timer expiration the related resource is deleted by "Delete the addressed resource".

b) If the sclRetrieve procedure completed unsuccessfully, the issuer SCL shall set the *aPocHandling* attribute to the value SHALLOW. The issuer SCL may retry the sending of the request starting from step 12, if it has reasons to believe that the failure was transient.

If the **receiver** is an DSCL or GSCL it shall execute the following steps:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

If the receiver is an NSCL then the **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting" The **receiver** shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource": Primitive specific operation:

   a) The Hosting SCL shall verify that the SCL-ID provided by the Issuer in the *sclId* attribute does not already exist.

   b) In case of verification failure and the provided SCL-ID already exists, the Hosting SCL shall reject the CREATE request, returning STATUS_CONFLICT.

   c) The Hosting SCL shall check if the mgmtProtocolType is supported. If not, it shall reject the CREATE request, returning STATUS_BAD_REQUEST.

3) Primitive specific operations: The hosting SCL may reject the request based on policies, which are not described in the present document. If the hosting SCL decides not to accept the request the request shall be rejected with a STATUS_PERMISSION_DENIED.

4) "Check the syntax of received message".

5) "Check validity of resource representation for CREATE".

6) Primitive specific operation: The **receiver** shall execute the following steps in order for integrity validation:

   a) If IVal is supported by the receiver SCL and the issuer SCL is IVal capable according to the information received during (clause 6) and IntegrityValResults are not included in the CREATE primitive, the hosting SCL shall reject the request with a STATUS_BAD_REQUEST.

   b) If IVal is supported by the receiver and the issuer is not IVal capable and IntegrityValResults are included in the CREATE primitive the hosting SCL shall reject the request with a STATUS_BAD_REQUEST.

   c) If IVal is supported by the receiver and the issuer is IVal Capable and IntegrityValResults are included in the CREATE primitive:

      i) The Ival key provided in the security attributes is used to validate the *IntegrityValResults* object. If the *IntegrityValResults* object is invalid, the receiver shall reject the request with a STATUS_BAD_REQUEST.

      ii) If the *IntegrityValResults* object is valid, the receiver shall use the integrity results therein to make a policy based access control decision.

      iii) If the result of the policy is to deny access, the receiver shall reject the request with a STATUS_FORBIDDEN.

   d) If IVal is not supported by the receiver, or if IVal is supported by the receiver and the result of the policy is to allow access, then continue with following steps.

7) "Create resource": Primitive specific operations:

   a) The Hosting SCL shall create a new *<scl>* resource structure under the *<sclBase>/scls/* with the uriEncoded value of the *sclId* attribute provided in the RequestIndication.

b) All the sub-resources defined as child of the <*scl*> Resource shall be created.

c) The resource attributes shall be initialized with the information provided by the issuer SCL. If no values are provided, the Hosting SCL shall use the default value.

d) The hosting SCL may set a value for the *aPocHandling* attribute based on operator policies. If the value is not set, then this shall be interpreted as if the *aPoCHandling* attribute is set to "SHALLOW".

8) "Create a successful ResponseConfirm".

9) "Send Response Confirm primitive".

10.6.2.2 sclCreateReponseConfirm(successful case)

This response is triggered by the sclCreateRequestIndication. The primitive shall comply with table 10.27. Optional parameters shall be returned in the response if hosting SCL over-wrote the settings that were specified in the request in order to bring the attributes in line with SCL policies.

Table 10.27: sclCreateResponseConfirm, successful case

SCL primitive: sclCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_CREATE_RESPONSE
statusCode	M	STATUS_CREATE
resourceURI	M	The URI of the scl resource that was created under <sclBase>/scls/
Resource	**Mandatory/Optional**	**Description**
scl	O	The scl resource created if Hosting SCL over-wrote the values provided in the request

10.6.2.3 sclCreateReponseConfirm(unsuccessful case)

This response is triggered by the sclCreateRequestIndication. The SCL primitive shall comply with table 10.28. Optional parameters shall be returned in the response if hosting SCL over-wrote the settings that were specified in the request in order to bring the attributes in line with SCL policies.

Table 10.28: sclCreateResponseConfirm, unsuccessful case

SCL primitive: sclCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_CREATE_RESPONSE
errorInfo	M	Provides Error information

10.6.3 sclRetrieve

10.6.3.1 sclRetrieveRequestIndication

This request is used for retrieving the content of an SCL resource. The SCL primitive shall comply with table 10.29.

Table 10.29: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Hosting SCL	Hosting SCL	Hosting SCL

Table 10.30: sclRetrieveRequestIndication

SCL Primitive: sclRetrieveRequestIndication		
Primitive Attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive.
targetID	M	This request shall address <sclBase>/scls/ of the target SCL.
primitiveType	M	SCL_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.6.3.2 sclRetrieveResponseConfirm (successful case)

This response is triggered by the sclRetrieveRequestIndication. The SCL primitive shall comply with table 10.31.

Table 10.31: sclRetrieveResponseConfirm, successful case

SCL primitive: sclRetrieveResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_RETREIVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
scl	M	Full representation of the *scl* resource, see clause 10.6.1

10.6.3.3 sclRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the sclRetrieveRequestIndication. The SCL primitive shall comply with table 10.32.

Table 10.32: sclRetrieveResponseConfirm, unsuccessful case

SCL primitive: sclRetrieveResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.6.4 sclUpdate

10.6.4.1 sclUpdateRequestIndication

This request is part of the SCL Update procedure. This request is used for updating one or more SCL attributes. The SCL primitive shall comply with table 10.33.

Table 10.33: Applicability of the primitive

SCL Primitive: sclUpdateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Hosting SCL	Hosting SCL	Hosting SCL

Table 10.34: sclUpdateRequestIndication

SCL Primitive: sclUpdateRequestIndication		
Primitive Attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the update of an scl resource
targetID	M	The URI of the target entity where the SCL resource shall be updated This request shall address <sclBase>/scls/ of the target SCL
primitiveType	M	SCL_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
scl	M	The resource representation of the *scl* to be updated see clause 10.6.1

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

Post Response operation:

1) Upon receiving a successful response, the **Issuer** SCL shall also update the new local *scl* resource, that represents the registered-to SCL as follows:

   a) If the ResponseConfirm contained a resource representation then the issuer SCL shall set the value of the expirationTime attribute to the value as received present in the resource representation in the ResponseConfirm.

   b) If the ResponseConfirm does not contain a resource representation then the issuer SCL shall set the value of the expirationTime attribute to the value as present in the resource representation in the sclUpdateRequestIndication.

2) If expirationTime attribute in the local *scl* resource is set to a non-negative time, then an expiration timer shall be re-started by the issuer SCL. At timer expiration the related resource is deleted by "Delete the addressed resource".

If the **receiver** is an DSCL or GSCL it shall execute the following steps:

3) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

4) "Send ResponseConfirm primitive".

If the **receiver** is an NSCL then the **receiver** shall execute the following steps in order.

In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of the resource representation for UPDATE".
6) "Update the addressed resource".
7) "Create a successful response Confirm".
8) "Send response Confirm primitive".

10.6.4.2 sclUpdateResponseConfirm (successful case)

This response is triggered by the sclUpdateRequestIndication. The SCL primitive shall comply with table 10.35. If the Hosting SCL did not accept the attribute values in the sclUpdateRequestIndication, then the attribute values that were used shall be included in the response.

Table 10.35: sclUpdateResponseConfirm, successful case

SCL primitive: sclUpdateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
scl	O	Full representation of the scl resource, if any of the provided attributes in the sclUpdateRequestIndication were modified by the hosting SCL, see clause 10.6.1

10.6.4.3 sclUpdateResponseConfirm (unsuccessful case)

This response is triggered by the sclUpdateRequestIndication. The SCL primitive shall comply with table 10.36. If the SCL did not accept the attribute values in the SclUpdateRequestIndication , then the attribute values that were used shall be included in the response.

Table 10.36: sclUpdateResponseConfirm, unsuccessful case

SCL primitive: sclUpdateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_UPDATE_RESPONSE
errorInfo	M	Provides Error information

10.6.5 sclDelete

10.6.5.1 sclDeleteRequestIndication

This request is part of the SCL delete procedure. This request is used for deleting SCL. The SCL primitive shall comply with table 10.37.

Table 10.37: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Hosting SCL	Hosting SCL	Hosting SCL

Table 10.38: sclDeleteRequestIndication

SCL Primitive: sclDeleteRequestIndication		
Primitive Attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL requesting the deletion
targetID	M	This request shall address the URI of the SCL to be deleted, e.g. <sclBase>/scls/ scl1
primitiveType	M	SCL_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

If the **receiver** is a DSCL or GSCL it shall execute the following steps.

1) Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

If the **receiver** is an NSCL then the **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Delete the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

Post response steps:

8) When the **issuer** receives the successful response from receiver scl, the issuer shall delete the local <scl> resource that it has created as part of the SCL registration procedure.

9) If the issuer of the DELETE request is not the creator of the remote <scl> at hosting scl, the following scenarios may occur:

   a) If the creator of the remote <scl> subscribed to the <scl> resource at the Hosting SCL, it shall be notified by the deletion of the <scl> resource at the Hosting SCL. As a consequence, the creator SCL shall delete the local <scl> it created corresponding to the registration.

   b) If the creator of the remote <scl> did not subscribe to the Hosting SCL, but the SCL registration period defined by expirationTime expires, the SCL shall perform a re-registration using the same procedure as the sclUpdate primitives defined above. A registration failure will occur in this case. Upon receiving the unsuccessful sclUpdateResponseConfirm, the SCL shall delete its local <scl> resource corresponding to this registration. Therefore, if the creator of the remote <scl> is not subscribed to the <scl> resource at the Hosting SCL, it shall not set the expirationTime as infinite.

10.6.5.2 sclDeleteResponseConfirm (successful case)

This response is triggered by the sclDeleteRequestIndication. The SCL primitive shall comply with table 10.39.

Table 10.39: sclDeleteResponseConfirm, successful case

SCL primitive: sclDeleteResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.6.5.3 sclDeleteResponseConfirm (unsuccessful case)

This response is triggered by the sclDeleteRequestIndication. The SCL primitive shall comply with table 10.40.

Table 10.40: sclDeleteResponseConfirm, unsuccessful case

SCL primitive: sclDeleteResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SCL_DELETE_RESPONSE
errorInfo	M	Provides error information

10.7 applications resource and management procedures

10.7.1 applications resource

The applications resource shall contain the following sub-resources and attributes.

Table 10.41: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
applicationCollection	N/A	NP	M	References to sub-resources (table 11.37)
applicationAnncCollection	N/A	NP	M	References to sub-resources (table 11.37)
subscriptionsReference	N/A	NP	M	Reference to sub-resource subscriptions in table 11.37
mgmtObjsReference	N/A	NP	M	Reference to sub-resource mgmtObjs in table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.7.2 applicationsCreate

The applications collection resource shall not be created directly via the API. It shall be created whenever the parent sclBase resource is created or when its parent scl resource is created. The accessRightID shall be initialized to the same value as the accessRightID in the parent.

Note that in case the applications resource is a child of an sclBase parent the creator of the sclBase may also initialize the scls resource with a accessRightID, e.g. according to internal configuration.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.7.3 applicationsRetrieve

10.7.3.1 applicationsRetrieveRequestIndication

This primitive is used to retrieve an applications Resource. The SCL primitive shall comply with the following.

Table 10.42: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.43: applicationsRetrieveRequestIndication

SCL Primitive: applicationsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieve
targetID	M	Indicates the applications resource to be retrieved
primitiveType	M	APPLICATIONS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the Receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of the received message".

5) "Create a collection resource representation".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.7.3.2 applicationsRetrieveResponseConfirm (successful case)

This primitive confirms the retrieval of an applications Resource. The SCL primitive shall comply with the following.

Table 10.44: ApplicationsRetrieveResponseConfirm, successful case

SCL primitive: applicationsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATIONS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
applications	M	The applications resource representation as indicated in clause 10.7.1

10.7.3.3 sclsRetrieveResponseConfirm (unsuccessful case)

This ResponseConfirm primitive is triggered by the applicationsRetrieveRequestIndication. The SCL primitive shall comply with table 10.45.

Table 10.45: ApplicationsRetrieveResponseConfirm, unsuccessful case

SCL primitive: applicationsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATIONS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.7.4 applicationsUpdate

10.7.4.1 applicationsUpdateRequestIndication

This primitive updates the applications resource. The SCL primitive shall comply with the following.

Table 10.46: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.47: ApplicationsUpdateRequestIndication

SCL Primitive: applicationsUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	URI of the applications resource to be updated
primitiveType	M	Indicates the APPLICATIONS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
applications	M	An applications resource representation as defined in clause 10.7.1 that replaces the current representation

The **issuer** shall execute the following steps in order:

1) "Compose a RequestIndication primitive".

2) "Send RequestIndication primitive to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of the received message".
5) "Check validity of resource representation for update".
6) "Update the addressed resource".
7) "Create successful ResponseConfirm".
8) "Send response Confirm primitive".

Note that since the resource representation only contains one Read-Write attribute (accessRightID), this shall be the only attribute that can be updated using this procedure.

10.7.4.2 applicationsUpdateResponseConfirm (successful case)

Successful response on updating of an existing applications Collection resource in a Service Capability Layer.

Table 10.48: ApplicationsUpdateRequestIndication, successful case

SCL Primitive: applicationsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATIONS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
applications	O	An applications resource full representation as clause 10.7.1. Only present if the hosting SCL modified any of the attributes provided by the Issuer.

10.7.4.3 applicationsUpdateResponseConfirm (unsuccessful case)

Unsuccessful response after an attempt to update an applications Collection resource in a Service Capability Layer.

Table 10.49: ApplicationsUpdateRequestIndication, unsuccessful case

SCL Primitive: applicationsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATIONS_UPDATE_RESPONSE
errorInfo	M	Provide error information

10.7.5 applicationsDelete

The applications resource shall not be deleted via the API.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.8 <application> resource and management procedures

10.8.1 <application> resource

The <application> resource shall contain the following attributes referring to sub-resources.

Table 10.50: Resource description

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
containersReference	NP	NP	M	Reference to sub-resource containers in table 11.37
groupsReference	NP	NP	M	Reference to sub-resource groups in table 11.37
accessRightsReference	NP	NP	M	Reference to sub-resource accessRights in table 11.37
subscriptionsReference	NP	NP	M	Reference to sub-resource subscriptions in table 11.37
notificationChannelsReference	NP	NP	M	Reference to sub-resource notificationChannels in table 11.37
appId	O	NP	M	The identity of the application, which is also used as the discriminator in the applications collection resource. The value shall be globally unique. If no value is provided in the CREATE request, the hosting SCL chooses an id. See clause 10.8.2
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
announceTo	O	O	M*	See table 11.36
aPoC	O	O	O	See table 11.36
aPoCPaths	O	O	O	See table 11.36
locrequester	O	O	O	See table 11.36

10.8.2 applicationCreate

10.8.2.1 applicationCreateRequestIndication

This procedure is used for registering and implicitly creating an application resource. The SCL primitive shall comply with table 10.51. The target SCL will use default values for optional parameters that are not specified.

Table 10.51: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	-
Receiver	Local SCL	Local SCL	-

Table 10.52: ApplicationCreateRequestIndication

SCL Primitive: applicationCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application requesting the creation of an application resource
targetID	M	The URI of the applications resource where the application resource shall be created This request shall address <sclBase>/applications/ of the target SCL
primitiveType	M	APPLICATION_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
application	M	The application resource representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) Primitive specific operation: if the issuer is provided with a globally unique ID (via out of band methods not described in the present document) then the issuer shall set this value on the id attribute in the application resource representation in the request. Otherwise, the issuer does not provide a unique ID in the *appId* attribute in the request, and a unique id shall be assigned by the receiver SCL.

3) "Send a RequestIndication to the receiver SCL".

4) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check the syntax of received message".

4) "Check validity of resource representation for CREATE".

5) "Create resource". Primitive specific operation: if no *appId* was received in the create request, then the issuer shall create a globally unique identifier and assign its value to the *appId* attribute of the application resource.

6) "Announce resource".

7) "Create a successful ResponseConfirm".

8) "Send Response Confirm primitive".

10.8.2.2 applicationCreateResponseConfirm (successful case)

This response is triggered by the ApplicationCreateRequestIndication. The SCL primitive shall comply with table 10.53.

Table 10.53: ApplicationCreateRequestConfirm, successful case

SCL primitive: applicationCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_CREATE_RESPONSE
statusCode	M	STATUS_OK
resourceURI	M	The URI of the created application resource
Resource	**Mandatory/Optional**	**Description**
application	O	Full representation of the application resource. Only present if any of the provided attributes were modified by the hosting SCL

10.8.2.3 applicationCreateResponseConfirm (unsuccessful case)

This response is triggered by the ApplicationCreateRequestIndication. The SCL primitive shall comply with table 10.54.

Table 10.54: ApplicationCreateRequestConfirm, unsuccessful case

SCL primitive: applicationCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_CREATE_RESPONSE
errorInfo	M	Provides Error information

10.8.3 applicationRetrieve

10.8.3.1 applicationRetrieveRequestIndication

This request is used for retrieving the content of an application resource. The SCL primitive shall comply with table 10.55.

Table 10.55: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.56: ApplicationRetrieveRequestIndication

SCL Primitive: ApplicationRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL requesting the retrieval
targetID	M	This request shall address resource URI of an application resource on the hosting SCL or the targetID shall address a URI that is hierarchically subordinate to resource URI of an <*application*> resource The latter is used in partial addressing see clause 10.39 or SCL retargeting (see below)
primitiveType	M	APPLICATION_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "SCL retargeting to an application" The receiver shall not execute following steps if the request is re-targeted.

4) "Check authorization of the requestingEntity based on accessRightID".

5) "Check the syntax of received message".

6) "Read the addressed resource". Primitive specific operations: If the *requestingEntity* is not the creator of the <*application*> resource, then the hosting SCL shall remove the *aPoC* attribute from the resource representation.

NOTE: This means that nobody but the Application corresponding to the <*application*> resource can read the value of the *aPoC* attribute using the API.

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.8.3.2 applicationRetrieveResponseConfirm (successful case)

This response is triggered by the ApplicationRetrieveRequestIndication. The SCL primitive shall comply with table 10.57.

Table 10.57: ApplicationRetrieveResponseConfirm, successful case

SCL primitive: applicationRetrieveResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
application	M	The retrieved application resource representation

10.8.3.3 applicationRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the ApplicationRetrieveRequestIndication. The SCL primitive shall comply with table 10.58.

Table 10.58: ApplicationRetrieveResponseConfirm, unsuccessful case

SCL primitive: applicationRetrieveResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.8.4 applicationUpdate

10.8.4.1 applicationUpdateRequestIndication

This request is part of the Application Update procedure. This request is used for updating one or more application attributes. The SCL primitive shall comply with table 10.59.

Table 10.59: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	-
Receiver	Local SCL	Local SCL	-

Table 10.60: ApplicationUpdateRequestIndication

SCL Primitive: applicationUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application requesting the update
targetID	M	This request shall address resource URI of an application resource on the hosting SCL or the targetID shall address a URI that is hierarchically subordinate to resource URI of an <*application*> resource The latter is used in partial addressing see clause 10.39 or SCL retargeting (see below)
primitiveType	M	APPLICATION_UPDATE_REQUEST
Resource	**Mandatory/optional**	**Description**
application	M	the resource representation as defined in above that replaces the current representation

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "SCL retargeting to an application" The receiver shall not execute following steps if the request is re-targeted.

4) "Check authorization of the requestingEntity based on accessRightID".

5) "Check the syntax of received message".

6) "Check validity of the resource representation for UPDATE".

7) "Update the addressed resource".

8) "Announce resource".

9) Primitive specific operations: If the *requestingEntity* is not the creator of the <*application*> resource, then the hosting SCL shall remove the *aPoC* attribute from the resource representation in the response.

NOTE: This means that nobody but the Application corresponding to the <*application*> resource can read the value of the *aPoC* attribute using the API.

10) "Create a successful response Confirm".

11) "Send response Confirm primitive".

10.8.4.2 applicationUpdateResponseConfirm (successful case)

This response is triggered by the ApplicationUpdateRequestIndication. The SCL primitive shall comply with table 10.61. If the SCL did not accept the attribute values in the ApplicationUpdateRequestIndication, then the attribute values that were used shall be included in the response.

Table 10.61: ApplicationUpdateRequestIndication, successful case

SCL primitive: applicationUpdateResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
application	O	Full representation of the application resource. Only present if any of the provided attributes were modified by the hosting SCL

10.8.4.3 applicationUpdateResponseConfirm (unsuccessful case)

This response is triggered by the ApplicationUpdateRequestIndication. The SCL primitive shall comply with table 10.62.

Table 10.62: applicationUpdateRequestIndication, unsuccessful case

SCL primitive: applicationUpdateResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_UPDATE_RESPONSE
errorInfo	M	Provides Error information

10.8.5 applicationDelete

10.8.5.1 applicationDeleteRequestIndication

The procedure is used for Delete an application resource. The SCL primitive shall comply with table 10.63.

Table 10.63: Applicability of the primitive

SCL Primitive: applicationDeleteRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	hosting SCL

Table 10.64: ApplicationDeleteRequestIndication

SCL Primitive: applicationDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL requesting the de-registration
targetID	M	This request shall address resource URI of an application resource on the hosting SCL
primitiveType	M	APPLICATION_DELETE_REQUEST

The **issuer** shall execute the following steps in order.

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Delete the addressed resource".
6) "DeAnnounce resource".
7) "Delete the addressed resource".
8) "Create a successful ResponseConfirm".
9) "Send ResponseConfirm primitive".

10.8.5.2 applicationDeleteResponseConfirm (successful case)

This response is triggered by the ApplicationDeleteRequestIndication. The SCL primitive shall comply with table 10.65.

Table 10.65: ApplicationDeleteRequestIndication, successful case

SCL primitive: applicationDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.8.5.3 applicationDeleteResponseConfirm (unsuccessful case)

This response is triggered by the ApplicationDeleteRequestIndication. The SCL primitive shall comply with table 10.66.

Table 10.66: ApplicationDeleteRequestIndication, unsuccessful case

SCL primitive: applicationDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_DELETE_RESPONSE
errorInfo	M	Provides Error information

10.9 *<applicationAnnc>* resource and management procedures

10.9.1 <applicationAnnc> resource

The *<applicationAnnc>* resource shall contain the following sub-resources and attributes.

Table 10.67: Resource description

AttributeName	Presence in createReq	Presence in updateReq\	Presence in response	Description
containersReference	NP	NP	M	See table 11.37
groupsReference	NP	NP	M	See table 11.37
accessRightsReference	NP	NP	M	See table 11.37
link	M	NP	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	M	M	M	See table 11.36
expirationTime	O	O	M*	See table 11.36
id	O	NP	M*	The ID of the announcement resource. This is used to identify the resource in its parents collection. The id is the id of the corresponding <application> resource postfixed with "Annc"

10.9.2 applicationAnncCreate

10.9.2.1 applicationAnncCreateRequestIndication

This request is used to create a new *<applicationAnnc>* resource in a Service Capability Layer. The SCL primitive shall comply with table 10.68.

Table 10.68: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.69: ApplicationAnncCreateRequestIndication

SCL Primitive: applicationAnncCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created
primitiveType	M	APPLICATION_ANNC_CREATE_REQUEST
Id	M	Indicates the identifier of the resource proposed to hosting SCL
Resource	**Mandatory/Optional**	**Description**
applicationAnnc	O	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "create announced resource".

10.9.2.2 applicationAnncCreateResponseConfirm (successful case)

It confirms the creation of a new <*applicationAnnc*> resource in a Service Capability Layer. The SCL primitive shall comply with table 10.70.

Table 10.70: ApplicationAnncCreateResponseConfirm, successful case

SCL primitive: applicationAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
applicationAnnc	O	In the case that **any** of the provided attributes in the request have been modified by the hosting SCL then the complete content of the resource as described above is returned in the response as well

10.9.2.3 applicationAnncCreateResponseConfirm (unsuccessful case)

This response is triggered by the *applicationAnncCreateRequestIndication* primitive. The SCL primitive shall comply with table 10.71.

Table 10.71: ApplicationAnncCreateResponseConfirm, unsuccessful case

SCL primitive: applicationAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.9.3 applicationAnncRetrieve

10.9.3.1 applicationAnncRetrieveRequestIndication

This request is used for retrieving the content of an <*applicationAnnc*> resource. The SCL primitive shall comply with table 10.72.

Table 10.72: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.73: ApplicationAnncRetrieveRequestIndication

SCL Primitive: applicationAnncRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	APPLICATION_ANNC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "retrieve announced resource".

10.9.3.2 applicationAnncRetrieveResponseConfirm (successful case)

This response is triggered by the *applicationAnncRetrieveRequestIndication* primitive. The SCL primitive shall comply with table 10.74.

Table 10.74: ApplicationAnncRetrieveResponseConfirm, successful case

SCL primitive: ApplicationAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
applicationAnnc	M	The resource representation is returned in the response

10.9.3.3 applicationAnncRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the *applicationAnncRetrieveRequestIndication* primitive. The SCL primitive shall comply with table 10.75.

Table 10.75: ApplicationAnncRetrieveResponseConfirm, unsuccessful case

SCL primitive: applicationAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.9.4 applicationAnncUpdate

10.9.4.1 applicationAnncUpdateRequestIndication

This request is used to update and to modify the *<applicationAnnc>* resource. The SCL primitive shall comply with table 10.76.

Table 10.76: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.77: ApplicationAnncUpdateRequestIndication

SCL Primitive: applicationAnncUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	APPLICATION_ANNC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
applicationAnnc	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "update announced resource".

10.9.4.2 applicationAnncUpdateResponseConfirm (successful case)

This response is triggered by the *applicationAnncUpdateRequestIndication* primitive. The SCL primitive shall comply with table 10.78.

Table 10.78: ApplicationAnncUpdateResponseConfirm, successful case

SCL primitive: applicationAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
applicationAnnc	O	Full representation of the retrieved announced resource

10.9.4.3 applicationAnncUpdateResponseConfirm (unsuccessful case)

This response is triggered by the *applicationAnncUpdateRequestIndication* primitive. The SCL primitive shall comply with table 10.79.

Table 10.79: ApplicationAnncUpdateResponseConfirm, unsuccessful case

SCL primitive: applicationAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.9.5 applicationAnncDelete

10.9.5.1 applicationAnncDeleteRequestIndication

The procedure is used to delete an <*applicationAnnc*> resource. The SCL primitive shall comply with table 10.80.

Table 10.80: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.81: ApplicationAnncDeleteRequestIndication

SCL Primitive: applicationAnncDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the announced resource deletion
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	APPLICATION_ANNC_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "delete announced resource".

10.9.5.2 applicationAnncDeleteResponseConfirm (successful case)

This response is triggered by the applicationAnncDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.82.

Table 10.82: ApplicationAnncDeleteResponseConfirm, successful case

SCL primitive: applicationAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.9.5.3 applicationAnncDeleteResponseConfirm (unsuccessful case)

This response is triggered by the applicationAnncDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.83.

Table 10.83: ApplicationAnncDeleteResponseConfirm, unsuccessful case

SCL primitive: applicationAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	APPLICATION_ANNC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.10 accessRights resource and management procedures

10.10.1 accessRights resource

The accessRights resource shall contain the following sub-resources and attributes.

Table 10.84: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
accessRightCollection	N/A	NP	M	references of accessRight sub-resources
accessRightAnncCollection	N/A	NP	M	references of accessRightsAnnc sub-resources
subscriptionsReference	N/A	NP	M	URI of subscription collection
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.10.2 accessRightsCreate

The accessRights resource shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.10.3 accessRightsRetrieve

10.10.3.1 accessRightsRetrieveRequestIndication

This request is used for retrieving the content of an access right collection resource. The SCL primitive shall comply with table 10.85.

Table 10.85: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.86: AccessRightsRetrieveRequestIndication

SCL Primitive: accessRightsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The resource URI of the accessRights resource to be retrieved
primitiveType	M	ACCESS_RIGHTS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Create a collection resource representation".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.10.3.2 accessRightsRetrieveResponseConfirm (successful case)

This response is triggered by the accessRightsRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.87.

Table 10.87: AccessRightsRetrieveResponseConfirm, successful case

SCL primitive: accessRightsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHTS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRights	M	The complete resource representation of the accessRights is returned in the response

10.10.3.3 accessRightsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the AccessRightsRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.88.

Table 10.88: AccessRightsRetrieveResponseConfirm, unsuccessful case

SCL primitive: accessRightsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHTS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.10.4 accessRightsUpdate

10.10.4.1 accessRightsUpdateRequestIndication

This request is used to update and modify an access right collection resource, or one or more resource attributes. The SCL primitive shall comply with table 10.89.

Table 10.89: Applicability of the primitive

SCL Primitive: accessRightsUpdateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.90: AccessRightsUpdateRequestIndication

SCL Primitive: accessRightsUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the accessRights resource to be updated
primitiveType	M	ACCESS_RIGHTS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
accessRights	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive:

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of resource representation for update".
6) "Update the addressed resource".
7) "Create a successful ResponseConfirm".
8) "Send ResponseConfirm primitive".

10.10.4.2 accessRightsUpdateResponseConfirm (successful case)

This response is triggered by the AccessRightsUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.91.

Table 10.91: AccessRightsUpdateResponseConfirm, successful case

SCL primitive: AccessRightsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHTS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRights	O	Full resource representation of the updated access right collection resource. Only present if any of the provided attributes where modified by the hosting SCL

10.10.4.3 accessRightsUpdateResponseConfirm (unsuccessful case)

This response is triggered by the accessRightsUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.92.

Table 10.92: AccessRightsUpdateResponseConfirm, unsuccessful case

SCL primitive: accessRightsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHTS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.10.5 accessRightsDelete

The accessRights resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.11 accessRight Resource and Management Procedures

10.11.1 accessRight resource

The <accessRight> resource shall contain the following attributes referring to sub-resources and attributes.

Table 10.93: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
subscriptionsReference	NP	NP	M	URI of subscription collection
expirationTime	O	O	M*	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
announceTo	O	O	M*	See table 11.36
permissions	O	O	M	See table 11.36
selfPermissions	M	M	M	See table 11.36
id	O	NP	M*	The ID of the <accessRight> resource. This is used to identify the resource in its parents collection.

10.11.2 accessRightCreate

10.11.2.1 accessRightCreateRequestIndication

This request is used to create a new access right resource in a Service Capability Layer. The SCL primitive shall comply with table 10.94.

Table 10.94: Applicability of the primitive

SCL Primitive: accessRightCreateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.95: AccessRightCreateRequestIndication

SCL Primitive: accessRightCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the resource URI of the collection resource in which the requested resources shall be created
primitiveType	M	ACCESS_RIGHT_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
accessRight	O	Refer to the above table for accessRight resource definition

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource". In this case the addressed resource shall be the provided access right collection resource.
3) "Check authorization of the requestingEntity based on accessRightID". In this case the checked access rights are those of the addressed collection.
4) "Check the syntax of received message".
5) "Check validity of resource representation for create".
6) "Create resource".
7) "Announce the resource".
8) "Create a successful ResponseConfirm".
9) "Send ResponseConfirm primitive".

10.11.2.2 accessRightCreateResponseConfirm (successful case)

It confirms the creation of a new access right resource in a Service Capability Layer. The SCL primitive shall comply with table 10.96.

Table 10.96: AccessRightCreateResponseConfirm, successful case

SCL primitive: AccessRightCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
accessRight	O	In the case that **any** of the provided attributes in the request have been modified by the hosting SCL (e.g. announceTo or expirationTime), then the complete content of the resource as described above is returned in the response as well

10.11.2.3 accessRightCreateResponseConfirm (unsuccessful case)

This response is triggered by the AccessRightCreateRequestIndication primitive. The SCL primitive shall comply with the table 10.97.

Table 10.97: AccessRightCreateResponseConfirm, unsuccessful case

SCL primitive: accessRightCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_CREATE_RESPONSE
errorInfo	M	Provides error information

10.11.3 accessRightRetrieve

10.11.3.1 accessRightRetrieveRequestIndication

This request is used for retrieving the content of an access right resource. The SCL primitive shall comply with table 10.98.

Table 10.98: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.99: AccessRightRetrieveRequestIndication

SCL Primitive: accessRightRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieval of the resource
targetID	M	The resource URI of the <accessRight> resource to be retrieved
primitiveType	M	ACCESS_RIGHT_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on selfPermissions".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.11.3.2 accessRightRetrieveResponseConfirm (successful case)

This response is triggered by the accessRightRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.100.

Table 10.100: AccessRightRetrieveResponseConfirm, successful case

SCL primitive: accessRightRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRight	M	Complete resource representation is returned in the response

10.11.3.3 accessRightRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the accessRightRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.101.

Table 10.101: AccessRightRetrieveResponseConfirm, unsuccessful case

SCL primitive: accessRightRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.11.4 accessRightUpdate

10.11.4.1 accessRightUpdateRequestIndication

This request is used to update and to modify the access right resource, or one or more access right attributes. The SCL primitive shall comply with table 10.102.

Table 10.102: Applicability of the primitive

AccessRightUpdateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.103: AccessRightUpdateRequestIndication

SCL primitive: accessRightUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the update
targetID	M	The URI of the access right resource to be updated
primitiveType	M	ACCESS_RIGHT_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
accessRight	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on selfPermissions".

4) "Check the syntax of received message".

5) "Check validity of resource representation for update".

6) "Update the addressed resource".

7) "Announce the resource".

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.11.4.2 accessRightUpdateResponseConfirm (successful case)

This response is triggered by the accessRightUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.104.

Table 10.104: AccessRightRetrieveResponseConfirm, successful case

SCL primitive: accessRightRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRight	O	Full representation of the update access right resource, only present in case the hosting SCL modified any of the provided attributes

10.11.4.3 accessRightUpdateResponseConfirm (unsuccessful case)

This response is triggered by the accessRightUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.105.

Table 10.105: AccessRightRetrieveResponseConfirm, unsuccessful case

SCL primitive: accessRightRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.11.5 accessRightDelete

10.11.5.1 accessRightDeleteRequestIndication

The procedure is used to delete an access right resource. The SCL primitive shall comply with table 10.106.

Table 10.106: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.107: accessRightDeleteRequestIndication

SCL Primitive: accessRightDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the access right resource deletion
targetID	M	The resource URI of the <accessRight> resource to be deleted
primitiveType	M	ACCESS_RIGHT_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on selfPermissions".

4) "Check the syntax of received message".

5) "De-Announce the resource".

6) "Delete the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.11.5.2 accessRightDeleteResponseConfirm (successful case)

This response is triggered by the accessRightDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.108.

Table 10.108: AccessRightDeleteResponseConfirm, successful case

SCL primitive: accessRightDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.11.5.3 accessRightDeleteResponseConfirm (unsuccessful case)

This response is triggered by the accessRightDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.109.

Table 10.109: AccessRightDeleteResponseConfirm, unsuccessful case

SCL primitive: accessRightDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_DELETE_RESPONSE
errorInfo	M	Provides error information

10.12 <accessRightAnnc> resource and management procedures

10.12.1 <accessRightAnnc> resource

The <accessRightAnnc> resource shall contain the following attributes referring to sub-resources.

Table 10.110: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
link	M	NP	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	M	M	M	See table 11.36
expirationTime	O	O	M*	See table 11.36
id	O	NP	M*	The ID of the announcement resource. This is used to identify the resource in its parents collection. The id is the id of the corresponding <accessRight> resource postfixed with "Annc"

10.12.2 accessRightAnncCreate

10.12.2.1 accessRightAnncCreateRequestIndication

This request is used to create a new <accessRightAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with table 10.111.

Table 10.111: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.112: accessRightAnncCreateRequestIndication

SCL Primitive: accessRightAnncCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the resource URI of collection resource in which the requested resource shall be created
primitiveType	M	ACCESS_RIGHT_ANNC_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
accessRightAnnc	O	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "create announced resource".

10.12.2.2 accessRightAnncCreateResponseConfirm (successful case)

It confirms the creation of a new <accessRightAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with table 10.113.

Table 10.113: AccessRightAnncCreateResponseConfirm, successful case

SCL primitive: accessRightAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource.
Resource	**Mandatory/Optional**	**Description**
accessRightAnnc	O	In the case that **any** of the provided attributes in the request have been modified by the hosting SCL then the complete content of the resource as described above is returned in the response as well

10.12.2.3 accessRightAnncCreateResponseConfirm (unsuccessful case)

This response is triggered by the accessRightAnncCreateRequestIndication primitive. The SCL primitive shall comply with table 10.114.

Table 10.114: accessRightAnncCreateResponseConfirm, unsuccessful case

SCL primitive: accessRightAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.12.3 accessRightAnncRetrieve

10.12.3.1 accessRightAnncRetrieveRequestIndication

This request is used for retrieving the content of an <accessRightAnnc> resource. The SCL primitive shall comply with table 10.115.

Table 10.115: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.116: AccessRightAnncRetrieveRequestIndication

SCL Primitive: accessRightAnncRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the announced resource to be addressed.
primitiveType	M	ACCESS_RIGHT_ANNC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "retrieve announced resource".

10.12.3.2 accessRightAnncRetrieveResponseConfirm (successful case)

This response is triggered by the accessRightAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.117.

Table 10.117: AccessRightAnncRetrieveResponseConfirm, successful case

SCL primitive: accessRightAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRightAnnc	M	Complete resource representation of the resource is returned in the response

10.12.3.3 accessRightAnncRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the accessRightAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.118.

Table 10.118: accessRightAnncRetrieveResponseConfirm, unsuccessful case

SCL primitive: accessRightAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.12.4 accessRightAnncUpdate

10.12.4.1 accessRightAnncUpdateRequestIndication

This request is used to update and to modify the <accessRightAnnc> resource. The SCL primitive shall comply with table 10.119.

Table 10.119: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.120: AccessRightAnncUpdateRequestIndication

SCL Primitive: accessRightAnncUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	ACCESS_RIGHT_ANNC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
accessRightAnnc	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "update announced resource".

10.12.4.2 accessRightAnncUpdateResponseConfirm (successful case)

This response is triggered by the accessRightAnncUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.121.

Table 10.121: AccessRightAnncUpdateResponseConfirm, successful case

SCL primitive: accessRightAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
accessRightAnnc	O	Full representation of the retrieved announced resource

10.12.4.3 accessRightAnncUpdateResponseConfirm (unsuccessful case)

This response is triggered by the AccessRightAnncUpdateRequestIndication primitive. The SCL primitive shall comply with table 10.122.

Table 10.122: AccessRightAnncUpdateResponseConfirm, unsuccessful case

SCL primitive: AccessRightAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.12.5 accessRightAnncDelete

10.12.5.1 accessRightAnncDeleteRequestIndication

The procedure is used to delete an <accessRightAnnc> resource. The SCL primitive shall comply with table 10.123.

Table 10.123: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.124: AccessRightAnncDeleteRequestIndication

SCL Primitive: accessRightAnncDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the announced resource deletion
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	ACCESS_RIGHT_ANNC_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "delete announced resource".

10.12.5.2 accessRightAnncDeleteResponseConfirm (successful case)

This response is triggered by the accessRightAnncDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.125.

Table 10.125: AccessRightAnncDeleteResponseConfirm, successful case

SCL primitive: accessRightAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.12.5.3 accessRightAnncDeleteResponseConfirm (unsuccessful case)

This response is triggered by the AccessRightAnncDeleteRequestIndication primitive. The SCL primitive shall comply with table 10.126.

Table 10.126: AccessRightAnncDeleteResponseConfirm, unsuccessful case

SCL primitive: accessRightAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ACCESS_RIGHT_ANNC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.13 containers resource and management procedures

10.13.1 containers resource

The containers resource shall contain the following sub-resources and attributes.

Table 10.127: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
containerCollection	N/A	NP	M	See table 11.37
containerAnncCollection	N/A	NP	M	See table 11.37
locationCollection	N/A	NP	M	See table 11.37
locationAnncCollection	N/A	NP	M	See table 11.37
subscriptionsReference	N/A	NP	M	See table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.13.2 containersCreate

The containers resource shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.13.3 containersRetrieve

10.13.3.1 containersRetrieveRequestIndication

This request is used to retrieve the content of a containers resource. The SCL primitive shall comply with table 10.128.

Table 10.128: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.129: ContainersRetrieveRequestIndication

SCL Primitive: containersRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the containers resource to be addressed This URI shall be one of the following: <sclBase>/containers <sclBase>/applications<application>/containers <sclBase>/scls/<scl>/containers <sclBase>/scls/<scl>/applications/<application>/containers If an attribute has to be retrieved, then the URI of the attribute shall be provided
primitiveType	M	CONTAINERS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of the received message".

5) "Read the addressed resource".

6) "Create a collection resource representation".

7) "Create successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.13.3.2 containersRetrieveResponseConfirm (successful case)

This response is triggered by the ContainersRetrieveRequestIndication. The SCL primitive shall comply with table 10.130.

Table 10.130: ContainersRetrieveResponseConfirm, successful case

SCL primitive: containersRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINERS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
containers	M	List of references(URLs)to all the child resources of the retrieved containers resource, and all the attributes defined

10.13.3.3 containersRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the containersRetrieveRequestIndication. The SCL primitive shall comply with table 10.131.

Table 10.131: ContainersRetrieveResponseConfirm, unsuccessful case

SCL primitive: containersRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINERS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.13.4 containersUpdate

10.13.4.1 containersUpdateRequestIndication

This request is used to retrieve the content of a containers resource. The SCL primitive shall comply with table 10.132.

Table 10.132: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.133: Applicability of the primitive

SCL Primitive: containersUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the containers resource to be addressed This URI shall be one of the following: <sclBase>/containers <sclBase>/applications<application>/containers <sclBase>/scls/<scl>/containers <sclBase>/scls/<scl>/applications/<application>/containers If an attribute has to be updated, then the URI of the attribute shall be provided
primitiveType	M	CONTAINERS_UPDATE_REQUEST
Resource Attribute	**Mandatory/Optional**	**Description**
containers	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".
2) "Send a RequestIndication to the receiver SCL".
3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of resource representation for update".
6) "Update the addressed resource".
7) "Read the addressed resource".
8) "Create a collection resource representation".
9) "Create a successful ResponseConfirm".
10) "Send ResponseConfirm primitive".

10.13.4.2 containersUpdateResponseConfirm (successful case)

This response is triggered by the ContainersUpdateRequestIndication. The SCL primitive shall comply with table 10.134.

Table 10.134: ContainersUpdateResponseConfirm, successful case

SCL primitive: containersUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINERS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
containers	O	Full representation of the updated containers resource

10.13.4.3 containersUpdateResponseConfirm (unsuccessful case)

This response is triggered by the containersUpdateRequestIndication. The SCL primitive shall comply with table 10.135.

Table 10.135: ContainersUpdateResponseConfirm, unsuccessful case

SCL primitive: ContainersUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINERS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.13.5 containersDelete

The containers resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.14 <container> resource and management procedures

10.14.1 <container> resource

The <container> resource shall contain the following sub-resources and attributes.

Table 10.136: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
contentInstancesReference	NP	NP	M	References to sub-resources contentInstances in table 11.37
subscriptionsReference	NP	NP	M	Reference to sub-resource subscriptions in table 11.37
id	O	NP	M*	Id of the container in the containers collection. If the container indicated in the request already exists, the hosting SCL may choose a different name
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
announceTo	O	O	M*	See table 11.36
maxNrOfInstances	O	O	M*	See table 11.36
maxByteSize	O	O	M*	See table 11.36
maxInstanceAge	O	O	M*	See table 11.36

10.14.2 containerCreate

10.14.2.1 containerCreateRequestIndication

This primitive creates a new <container> resource in a containers collection. The SCL primitive shall comply with table 10.137.

Table 10.137: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	SCL	SCL	Hosting SCL

Table 10.138

SCL Primitive: containerCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	Indicates the type of primitive: CONTAINER_CREATE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to create a container resource
targetID	M	The URI of the target entity This URI shall be one of the following: <sclBase>/containers <sclBase>/applications/<application>/containers <sclBase>/scls/<scl>/containers <sclBase>/scls/<sclName>/applications/<applicationAnnc>/containers
Resource Attribute	**Mandatory/Optional**	**Description**
container	M	The container resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

6) "Create the resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.14.2.2 containerCreateResponseConfirm (successful case)

This primitive confirms the creation of a new <container> resource in a Service Capability Layer. The SCL primitive shall comply with table 10.139.

Table 10.139: ContainerCreateResponseConfirm, successful case

SCL primitive: containerCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/optional**	**Description**
container	O	Full resource representation of the container. This is only present if any of the provided attributes where modified by the hosting SCL

10.14.2.3 containerCreateResponseConfirm (unsuccessful case)

This primitive confirms the creation of a new <container> resource in a Service Capability Layer. The SCL primitive shall comply with table 10.140.

Table 10.140: ContainerCreateResponseConfirm, unsuccessful case

SCL primitive: containerCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_CREATE_RESPONSE
errorInfo	M	Provides error information

10.14.3 containerRetrieve

10.14.3.1 containerRetrieveRequestIndication

This primitive is used to read the attributes of a <container> and references to its direct child resources. The SCL primitive shall comply with table 10.141.

Table 10.141: Applicability of the primitive

SCL Primitive: containerRetrieveRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Hosting SCL	Hosting SCL	Hosting SCL

Table 10.142: ContainerRetrieveRequestIndication

SCL Primitive: containerRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_RETRIEVE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to read the content of a container resource
targetID	M	The URI of the container resource to be addressed

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send response ResponseConfirm primitive".

10.14.3.2 containerRetrieveResponseConfirm (successful case)

If the entire container was addressed, then a representation of the entire resource will be returned. If only a single attribute was addressed, then only the attribute value is returned. The SCL primitive shall comply with table 10.143.

Table 10.143: ContainerRetrieveResponseConfirm, successful case

SCL primitive: containerRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
container	M	Full representation of the retrieved container resource

10.14.3.3 containerRetrieveResponseConfirm (unsuccessful case)

The SCL primitive shall comply with table 10.144.

Table 10.144: ContainerRetrieveResponseConfirm, unsuccessful case

SCL primitive: containerRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.14.4 containerUpdate

10.14.4.1 containerUpdateRequestIndication

This request is used to update all of the attributes in a <container>. The SCL primitive shall comply with table 10.145.

Table 10.145: Applicability of the primitive

SCL Primitive: containerUpdateRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.146: ContainerUpdateRequestIndication

SCL Primitive: containerUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_UPDATE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to update a container resource
targetID	M	The URI of the group resource to be addressed If an attribute has to be updated, then the URI of the attribute shall be provided
Resource Attribute	**Mandatory/Optional**	**Description**
container	M	The resource representative

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for UPDATE".

6) Primitive specific operation: if the maxNrOfInstances, maxByteSize or maxInstanceAge are changed in the <container> resource, the hosting SCL shall immediately enforce the new restrictions. The hosting SCL may check whether the instances of the container resource are still within these new restrictions. If the maxima are exceeded, the oldest <contentInstance> resources may be deleted.

7) "Update the addressed resource".

8) "Announce resource".

9) "Create a successful response Confirm".

10) "Send response Confirm primitive".

10.14.4.2 containerUpdateResponseConfirm (successful case)

If the requested values were overwritten by SCL policies, then the new attribute values shall be included in the response. The SCL primitive shall comply with table 10.147.

Table 10.147: ContainerUpdateResponseConfirm, successful case

SCL primitive: containerUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
container	O	Full representation of the updated container resource, if any of the provided attributes were modified by the hosting SCL

10.14.4.3 containerUpdateResponseConfirm (unsuccessful case)

SCL primitive shall comply with table 10.148.

Table 10.148: ContainerUpdateResponseConfirm, unsuccessful case

SCL primitive: containerUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.14.5 containerDelete

10.14.5.1 containerDeleteRequestIndication

This request is used to delete a <container> resource. The SCL primitive shall comply with table 10.149.

Table 10.149: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	SCL	SCL	Hosting SCL

Table 10.150: ContainerDeleteRequestIndication

SCL Primitive: containerDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_DELETE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to delete a container resource
targetID	M	The URI of the target <container> to be deleted

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Delete the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.14.5.2 containerDeleteResponseConfirm(successful case)

The SCL primitive shall comply with the following.

Table 10.151: ContainerDeleteResponseConfirm, successful case

SCL primitive: containerDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.14.5.3 containerDeleteResponseConfirm (unsuccessful case)

This response is issued if the containerDeleteRequestIndication was not successfully serviced. The SCL primitive shall comply with the following.

Table 10.152: ContainerDeleteResponseConfirm, unsuccessful case

SCL primitive: containerDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_DELETE_RESPONSE
errorInfo	M	Provides error information

10.15 <containerAnnc> resource and management procedures

10.15.1 <containerAnnc> resource

The <containerAnnc> resource shall contain the following attributes referring to sub-resources.

Table 10.153: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
link	M	NP	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	M	M	M	See table 11.36
expirationTime	O	O	M*	See table 11.36
id	O	NP	M*	The ID of the announcement resource. This is used to identify the resource in its parents collection. The id is the id of the corresponding <container> resource postfixed with "Annc"

10.15.2 containerAnncCreate

10.15.2.1 containerAnncCreateRequestIndication

This request is used to delete a <containerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.154: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.155: ContainerAnncCreateRequestIndication

SCL Primitive: containerAnncCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created
primitiveType	M	CONTAINER_ANNC_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
containerAnnc	M	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Create announced resource".

10.15.2.2 containerAnncCreateResponseConfirm (successful case)

It confirms the creation of a new <containerAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.156: ContainerAnncCreateResponseConfirm, successful case

SCL primitive: containerAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource.
Resource	**Mandatory/Optional**	**Description**
containerAnnc	O	In the case that **any** of the provided attributes in the request have been modified by the hosting SCL then the complete content of the resource as described above is returned in the response as well

10.15.2.3 containerAnncCreateResponseConfirm (unsuccessful case)

This response is triggered by the containerAnncCreateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.157: ContainerAnncCreateResponseConfirm, unsuccessful case

SCL primitive: containerAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.15.3 containerAnncRetrieve

10.15.3.1 containerAnncRetrieveRequestIndication

This request is used for retrieving the content of an <containerAnnc> resource. The SCL primitive shall comply with table 10.158.

Table 10.158: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.159: ContainerAnncRetrieveRequestIndication

SCL Primitive: containerAnncRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	CONTAINER_ANNC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Retrieve announced resource".

10.15.3.2 containerAnncRetrieveResponseConfirm (successful case)

This response is triggered by the ContainerAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.160

Table 10.160: ContainerAnncRetrieveResponseConfirm, successful case

SCL primitive: containerAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
containerAnnc	M	Complete content of the resource is returned in the response

10.15.3.3 containerAnncRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the containerAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.161.

Table 10.161: ContainerAnncRetrieveResponseConfirm, unsuccessful case

SCL primitive: containerAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.15.4 containerAnncUpdate

10.15.4.1 containerAnncUpdateRequestIndication

This request is used to update and to modify the <containerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.162: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.163: ContainerAnncUpdateRequestIndication

SCL Primitive: containerAnncUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	CONTAINER_ANNC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Descrption**
containerAnnc	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Update announced resource".

10.15.4.2 containerAnncUpdateResponseConfirm (successful case)

This response is triggered by the containerAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.164: ContainerAnncUpdateResponseConfirm, successful case

SCL primitive: containerAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
containerAnnc	O	Full representation of the retrieved announced resource

10.15.4.3 containerAnncUpdateResponseConfirm (unsuccessful case)

This response is triggered by the containerAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.165: ContainerAnncUpdateResponseConfirm, unsuccessful case

SCL primitive: containerAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.15.5 containerAnncDelete

10.15.5.1 containerAnncDeleteRequestIndication

The procedure is used to delete an <containerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.166: Applicability of the primitive

SCL Primitive: containerAnncDeleteRequestIndication			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.167: ContainerAnncDeleteRequestIndication

SCL Primitive: containerAnncDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the announced resource deletion
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	CONTAINER_ANNC_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Delete announced resource".

10.15.5.2 containerAnncDeleteResponseConfirm (successful case)

This response is triggered by the containerAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.168: ContainerAnncDeleteResponseConfirm, successful case

SCL primitive: containerAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.15.5.3 containerAnncDeleteResponseConfirm (unsuccessful case)

This response is triggered by the containerAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.169: ContainerAnncDeleteResponseConfirm, unsuccessful case

SCL primitive: containerAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTAINER_ANNC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.16 locationContainer resources and management procedures

10.16.1 <locationContainer> resource

The <locationContainer> resource shall contain the following sub-resources and attributes.

Table 10.170: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
contentInstancesReference	NP	NP	M	References to sub-resources contentInstances in table 11.37
subscriptionsReference	NP	NP	M	Reference to sub-resource subscriptions in table 11.37
id	O	NP	M*	Id of the locationContainer in the containers collection. If the locationContainer indicated in the request already exists, the hosting SCL may choose a different name
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
announceTo	O	O	M*	See table 11.36
maxNrOfInstances	O	O	M*	See table 11.36
maxByteSize	O	O	M*	See table 11.36
maxInstanceAge	O	O	M*	See table 11.36
locationContainerType	M	NP	M	See table 11.36

10.16.2 locationContainerCreate

10.16.2.1 locationContainerCreateRequestIndication

This primitive creates a new <locationContainer> resource in a containers collection. The SCL primitive shall comply with the following.

Table 10.171: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	Local SCL
Receiver	SCL	SCL	Hosting SCL

Table 10.172: LocationContainerCreateRequestIndication

SCL Primitive: locationContainerCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	The entity (an application or SCL) that requests to create a container resource
targetID	M	The URI of the target entity This URI shall be one of the following: <sclBase>/applications/<application>/containers <sclBase>/scls/<sclName>/applications/<applicationAnnc>/containers
primitiveType	M	LOCATION_CONTAINER_CREATE_REQUEST
Resource	**Mandatory/optional**	**Description**
locationContainer	M	locationContainer resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".
2) "Send the RequestIndication to the receiver SCL".
3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of resource representation for create".
6) Primitive specific operation: The hosting SCL shall check that the parent of the addresses containers resource is an <application> or <applicationAnnc> resource. If the parent is of another type, the request shall be rejected with a STATUS_FORBIDDEN.
7) Primitive specific operation: If the locationContainerType attribute in the received resource representation is "LOCATION_SERVER_BASED", the hosting SCL shall check the that the targetID is one of:

   - *<nsclBase>/scls/<dscl>/applications/<applicationAnnc of device>/containers.*
   - *<nsclBase>/scls/<gscl>/applications/<applicationAnnc of gateway>/containers.*

   This ensures that only containers resource residing under a D/G SCL registration resource in the NSCL can contain the <locationContainer> resource whose type attribute is "LOCATION_SERVER_BASED".

   If the targetID does not match one of these two templates, the request shall be rejected with a STATUS_FORBIDDEN.

8) "Create the resource".
9) "Announce the resource".
10) "Create a successful ResponseConfirm".
11) "Send ResponseConfirm primitive".

10.16.2.2 locationContainerCreateResponseConfirm (successful case)

This response is triggered by the locationContainerCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.173: LocationContainerCreateResponseConfirm (successful case)

SCL primitive: locationContainerCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_CREATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
locationContainer	O	Full representation of the created member resources. This is only present if any of the provided attributes where modified by the hosting SCL

10.16.2.3 locationContainerCreateResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.174: LocationContainerCreateResponseConfirm (unsuccessful case)

SCL primitive: locationContainerCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_CREATE_RESPONSE
errorInfo	M	Provides error information

10.16.3 locationContainerRetrieve

10.16.3.1 locationContainerRetrieveRequestIndication

This request is used for retrieving the content of a <locationContainer> resource. The SCL primitive shall comply with the following.

Table 10.175: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.176: LocationContainerRetrieveRequestIndication

SCL Primitive: locationContainerRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the locationContainer resource to be addressed If an attribute has to be retrieved, then the URI of the attribute shall be provided
primitiveType	M	Indicates the type of primitive: LOCATION_CONTAINER_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.16.3.2 locationContainerRetrieveResponseConfirm (successful case)

This response is triggered by the locationContainerRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.177: LocationContainerRetrieveResponseConfirm (successful case)

SCL primitive: locationContainerRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
locationContainer	M	Full representation of the retrieved locationContainer resource

10.16.3.3 locationContainerRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.178: LocationContainerRetrieveResponseConfirm (unsuccessful case)

SCL primitive: locationContainerRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.16.4 locationContainerUpdate

10.16.4.1 locationContainerUpdateRequestIndication

This request is used to update the full representation of the locationContainer resource. It is also possible to update only part of the locationContainer representation, see clause 10.39. The SCL primitive shall comply with the following.

Table 10.179: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.180: LocationContainerUpdateRequestIndication

SCL Primitive: locationContainerUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the locationContainer resource to be addressed If an attribute has to be updated, then the URI of the attribute shall be provided
primitiveType	M	LOCATION_CONTAINER_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
locationContainer	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for UPDATE".

6) Primitive specific operation: If the type attribute is changed from "APPLICATION_GENERATED" to "LOCATION_SERVER_BASED", it shall also check the <locationContainer> resource to find whether the locationContainer resource only reside under the containers resource, of which URIs are as below:

   - *<nsclBase>/scls/<dscl>/applications/<applicationAnnc of device >/containers.*

   - *<nsclBase>/scls/<gscl>/applications/<applicationAnnc of gateway >/containers.*

   This ensures that only containers resource residing under a D/G SCL registration resource in the NSCL can contain the <locationContainer> resource whose type attribute is "LOCATION_SERVER_BASED".

   If locationContainer doesn't reside under these kinds of container resources, then the request shall be rejected with a STATUS_FORBIDDEN.

7) "Update the addressed resource".

8) "Announce resource".

9) "Create a successful ResponseConfirm".

10) "Send ResponseConfirm primitive".

10.16.4.2 locationContainerUpdateResponseConfirm (successful case)

This response is triggered by the locationContainerUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.181: LocationContainerUpdateResponseConfirm (successful case)

SCL primitive: locationContainerUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_UPDATERESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
locationContainer	O	Full representation of the updated locationContainer resource, if any of the provided attributes were modified by the hosting SCL

10.16.4.3 locationContainerUpdateResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.182: LocationContainerUpdateResponseConfirm (unsuccessful case)

SCL primitive: locationContainerUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_UPDATERESPONSE
errorInfo	M	Provides error information

10.16.5 locationContainerDelete

10.16.5.1 locationContainerDeleteRequestIndication

The procedure is used to delete a <locationContainer> resource. The SCL primitive shall comply with the following.

Table 10.183: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.184: LocationContainerDeleteRequestIndication

SCL Primitive: locationContainerDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the locationContainer resource deletion
targetID	M	The URI of the locationContainer resource to be addressed
primitiveType	M	LOCATION_CONTAINER_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Delete the addressed resource".

6) "Deannounce resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.16.5.2 locationContainerDeleteResponseConfirm (successful case)

This response is triggered by the locationContainerDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.185: LocationContainerDeleteResponseConfirm (successful case)

SCL primitive: locationContainerDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.16.5.3 locationContainerDeleteResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.186: LocationContainerDeleteResponseConfirm (unsuccessful case)

SCL primitive: locationContainerDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_DELETE_RESPONSE
errorInfo	M	Provides error information

10.17 <locationcontainerAnnc> resource and management procedures

10.17.1 <locationContainerAnnc> resource

The <locationContainerAnnc> resource shall contain the following attributes referring to sub-resources.

Table 10.187: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
link	M	NP	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	M	M	M	See table 11.36
expirationTime	O	O	M*	See table 11.36
id	O	NP	M*	The ID of the announcement resource. This is used to identify the resource in its parents collection. The id is the id of the corresponding <locationContainer> resource postfixed with "Annc"

10.17.2 locationContainerAnncCreate

10.17.2.1 locationContainerAnncCreateRequestIndication

This request is used to create a new <locationContainerAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.188: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.189: LocationContainerAnncCreateRequestIndication

SCL Primitive: locationContainerAnncCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created
primitiveType	M	LOCATION_CONTAINER_ANNC_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
locationContainerAnnc	O	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Create announced resource".

10.17.2.2 locationContainerAnncCreateResponseConfirm (successful case)

It confirms the creation of a new <locationContainerAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.190: LocationContainerAnncCreateResponseConfirm (successful case)

SCL primitive: locationContainerAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
locationContainerAnnc	O	In the case that **any** of the provided attributes in the request have been modified by the hosting SCL then the complete content of the resource as described above is returned in the response as well

10.17.2.3 locationContainerAnncCreateResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerAnncCreateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.191: LocationContainerAnncCreateResponseConfirm (unsuccessful case)

SCL primitive: locationContainerAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.17.3 locationContainerAnncRetrieve

10.17.3.1 locationContainerAnncRetrieveRequestIndication

This request is used for retrieving the content of an <locationContainerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.192: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.193: LocationContainerAnncRetrieveRequestIndication

SCL Primitive: locationContainerAnncRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the announced resource to be addressed.
primitiveType	M	LOCATION_CONTAINER_ANNC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Retrieve announced resource".

10.17.3.2 locationContainerAnncRetrieveResponseConfirm (successful case)

This response is triggered by the locationContainerAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.194: LocationContainerAnncRetrieveResponseConfirm (successful case)

SCL primitive: locationContainerAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
locationContainerAnnc	M	Complete content of the resource is returned in the response

10.17.3.3 locationContainerAnncRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.195: LocationContainerAnncRetrieveResponseConfirm (unsuccessful case)

SCL primitive: locationContainerAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.17.4 locationContainerAnncUpdate

10.17.4.1 locationContainerAnncUpdateRequestIndication

This request is used to update and to modify the <locationContainerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.196: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.197: LocationContainerAnncUpdateRequestIndication

SCL Primitive: LocationContainerAnncUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	LOCATION_CONTAINER_ANNC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
locationContainerAnnc	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Update announced resource".

10.17.4.2 locationContainerAnncUpdateResponseConfirm (successful case)

This response is triggered by the LocationContainerAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.198: LocationContainerAnncUpdateResponseConfirm (successful case)

SCL primitive: LocationContainerAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
locationContainerAnnc	O	Full representation of the retrieved announced resource

10.17.4.3 locationContainerAnncUpdateResponseConfirm (unsuccessful case)

This response is triggered by the LocationContainerAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.199: LocationContainerAnncUpdateResponseConfirm (unsuccessful case)

SCL primitive: LocationContainerAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.17.5 locationContainerAnncDelete

10.17.5.1 locationContainerAnncDeleteRequestIndication

The procedure is used to delete an <locationContainerAnnc> resource. The SCL primitive shall comply with the following.

Table 10.200: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.201: locationContainerAnncDeleteRequestIndication

SCL Primitive: locationContainerAnncDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the announced resource deletion
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	LOCATION_CONTAINER_ANNC_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Delete announced resource".

10.17.5.2 locationContainerAnncDeleteResponseConfirm (successful case)

This response is triggered by the locationContainerAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.202: LocationContainerAnncDeleteResponseConfirm (successful case)

SCL primitive: locationContainerAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	***Description***
primitiveType	M	LOCATION_CONTAINER_ANNC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.17.5.3 locationContainerAnncDeleteResponseConfirm (unsuccessful case)

This response is triggered by the locationContainerAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.203: LocationContainerAnncDeleteResponseConfirm (unsuccessful case)

SCL primitive: locationContainerAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	LOCATION_CONTAINER_ANNC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.18 contentInstances resource and management procedures

10.18.1 contentInstances resource

The contentInstances resource shall contain the following attributes.

Table 10.204: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
contentInstanceCollection	N/A	NA	M*	Contains sub-resources <contentInstance> in table 11.37
latest	N/A	NA	O	References to the .last added <contentInstance> sub-resource. See table 11.36
oldest	N/A	NA	O	References to the .oldest remaining <contentInstance> sub-resource. See table 11.36
subscriptionsReference	N/A	NA	M*	References to sub-resource subscriptions in table 11.37
creationTime	N/A	NA	M*	See table 11.36
lastModifiedTime	N/A	NA	M*	See table 11.36
currentNrOfInstances	N/A	NA	M*	See table 11.36
currentByteSize	N/A	NA	M*	See table 11.36

10.18.2 contentInstancesCreate

The contentInstances resource shall be created as part of the parent and it shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.18.3 contentInstancesRetrieve

10.18.3.1 contentInstancesRetrieveRequestIndication

This request is used to read the content of a <contentInstances resource>. This request is used to get a list of data of all instances in the addressed contentInstances collection. The SCL primitive shall comply with the following.

Table 10.205: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.206: ContentInstancesRetrieveRequestIndication

SCL Primitive: contentInstancesRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the contentInstances resource to be addressed
primitiveType	M	CONTENT_INSTANCES_RETRIEVE_REQUEST
Resource attribute	**Mandatory/Optional**	**Description**
contentInstancesFilterCriteria	O	If no filterCriteria is specified, then all the data of instance resource shall be returned

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID". However, this case the accessRightID of the parent resource (the container resource) shall be used to check the authorization.

4) "Check the syntax of received message".

5) Primitive specific operation: Check the type of the parent resource:

   - If the parent resource is of type "locationContainer" then continue with then next step (step 6).

   - If the parent resource is of type "container" then continue with step 10.

6) Primitive specific operation: Check the *locationContainerType* attribute of the parent resource.

   - If the value of the *locationContainerType* attribute is "APPLICATION_GENERATED" continue with step 10.

   - Else continue with the next step.

7) **LocationContainer of type LOCATION_SERVER_BASED specific handling**
   Primitive specific operation: If filterCriteria exists, the hosting SCL shall reject the request with a STATUS_FORBIDDEN.

8) Primitive specific operation: The hosting SCL shall retrieve the *locTargetDevice* attribute from the <scl> grandparent of the addressed *contentInstances* resource (i.e. the <scl> resource under which the parent locationContainer resource resides).
   The hosting SCL shall also retrieve the *locRequestor* attribute from the resource corresponding to the requestingEntity (i.e. an <application> resource).
   If the requestingEntity is an SCL, the hosting SCL shall reject the request with a STATUS_FORBIDDEN.
   In case either the locTargetDevice or the locRequestor cannot be obtained, the hosting SCL shall reject the request with a STATUS_FORBIDDEN.
   Then the hosting SCL shall transform the received RequestIndication into LCS (LoCation Services) request, using the obtained locRequestor and locTargetDevice attributes.
   The hosting SCL shall provide default values for other attributes (e.g. location accuracy) required in the LCS service request (e.g. complying with Le interface defined in OMA Mobile Location Protocol [OMA MLP]) according to local policies.
   The hosting SCL shall send this LCS service request to the location server, e.g. through Le interface.

9) Primitive specific operation: The hosting SCL shall receive the corresponding LCS service response from the location server and transform it into a ResponseConfirm primitive. Note that the location server performs the privacy control and only responds successfully if the positioning procedure is permitted.
   If the LCS service response indicates the request was unsuccessful, then the hosting SCL shall reject the RequestIndication. The table below provides a statusCode mapping for the OMA MLS statusCodes. In case OMA MLS is used, then this mapping shall be applied. The type of error information for items7, 10-20, 24, 30-34 depends on the local policies. If NSCL will revise the LCS service request for the requestingEntity when receiving the future request from this requestingEntity, then the hosting SCL shall respond to the issuer with STATUS_INTERNAL_SERVER_ERROR. Otherwise, if the NSCL will not revise the future request, then the hosting SCL shall respond to the issuer with STATUS_BAD_REQUEST. This completes the retrieval procedure between hosting SCL and location server.
   Continue with step 12. Step 10 and 11 shall be skipped.

10) **Container or locationContainer of type APPLICATION_GENERATED**
"read the addressed resource". Primitive specific operation: the hosting SCL shall iterate over the children (contentInstance resources) and only include the resource representation of those children whose filter criteria match (see FilterCriteria in clause 11.4 for a definition of matching). in the resource representation of the returned contentInstances collection resource representation. If the metaDataOnly element of the filterCriteria exists in the request and if it is set to TRUE, then content element for contentInstances shall not be included. Otherwise, the content element will be included.

11) "Create a successful ResponseConfirm".

12) "Send ResponseConfirm primitive".

Table 10.207

No.	Status Code defined for OMA MLS [65]	Status Code in ResponseConfirm primitive
1	OK	STATUS_OK
2	SYSTEM FAILURE	STATUS_INTERNAL_SERVER_ERROR
3	UNSPECIFIED ERROR	STATUS_INTERNAL_SERVER_ERROR
4	UNAUTHORIZED APPLICATION	STATUS_INTERNAL_SERVER_ERROR
5	UNKNOWN SUBSCRIBER	STATUS_NOT_FOUND
6	ABSENT SUBSCRIBER	STATUS_NOT_FOUND
7	POSITION METHOD FAILURE	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
8	TIMEOUT	Not used
9	CONGESTION IN LOCATION SERVER	STATUS_SERVICE_UNAVAILABLE
10	UNSUPPORTED VERSION	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
11	TOO MANY POSITION ITEMS	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
12	FORMAT ERROR	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
13	SYNTAX ERROR	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
14	PROTOCOL ELEMENT NOT SUPPORTED	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
15	SERVICE NOT SUPPORTED	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
16	PROTOCOL ELEMENT ATTRIBUTE NOT SUPPORTED	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
17	INVALID PROTOCOL ELEMENT VALUE	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
18	INVALID PROTOCOL ELEMENT ATTRIBUTE VALUE	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
19	PROTOCOL ELEMENT VALUE NOT SUPPORTED	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
20	PROTOCOL ELEMENT ATTRIBUTE VALUE NOT SUPPORTED	STATUS_INTERNAL_SERVER_ERROR or STATUS_FORBIDDEN
21	CANCELLATION OF TRIGGERED LOCATION REQUEST	Not used
22	INVALID MSID IN TLRSR	Not used
23	TLRSR FOR INDIVIDUAL TARGET NOT SUPPORTED	Not used
24	QOP NOT ATTAINABLE	STATUS_INTERNAL_SERVER_ERROR or STATUS_BAD_REQUEST
25	POSITIONING NOT ALLOWED	STATUS_FORBIDDEN
26	CONGESTION IN MOBILE NETWORK	STATUS_SERVICE_UNAVAILABLE
27	DISALLOWED BY LOCAL REGULATIONS	STATUS_FORBIDDEN
28	MISCONFIGURATION OF LOCATION SERVER	STATUS_INTERNAL_SERVER_ERROR
29	TARGET MOVED TO NEW MSC/SGSN	Not used
30	STANDARD LOCATION REPORT SERVICE NOT SUPPORTED	Not used
31	MLS CLIENT ERROR	Not used
32	STANDARD LOCATION REPORT SERVICE NOT ACCEPTED	Not used
33	SUBSCRIBER IN STANDARD LOCATION REPORT SERVICE NOT VALID	Not used
34	INVALID SERVICE ID IN STANDARD LOCATION REPORT SERVICE	Not used

10.18.3.2 contentInstancesRetrieveResponseConfirm (successful case)

Returns a list of data instances in the addressed contentInstances collection or the meta-data of the data instances in the addressed contentInstances collection. The SCL primitive shall comply with the following.

Table 10.208: ContentInstancesRetrieveResponseConfirm (successful case)

SCL primitive: contentInstancesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCES_RETRIEVE_RESPONSE.
statusCode	M	STATUS_OK.
Resource	**Mandatory/Optional**	**Description**
contentInstances	M	It indicates one of the following possibilities: Full representation of the retrieved contentInstances resource with the data of all the <contentInstance> child resources. If the metaDataOnly member was set, then the metadata of all <contentInstance> is included in the response. The data shall not be contained in the response. Full representation of the retrieved contentInstances resource with the data of the <contentInstance> child resources that match the filterCriteria. If the metaDataOnly field was set, then the meta-data of the <contentInstance> that match the filterCriteria is included in the response. The data shall not be contained in the response. Full representation of the retrieved contentInstances resource with the location information received from location server.

10.18.3.3 contentInstancesRetrieveResponseConfirm (unsuccessful case)

SCL primitive shall comply with the following.

Table 10.209: ContentInstancesRetrieveResponseConfirm (unsuccessful case)

SCL primitive: contentInstancesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCES_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.18.4 contentInstancesUpdate

The <contentInstances> resource shall not be updated via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.18.5 contentInstancesDelete

The contentInstances resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.19 <contentInstance> resource and management procedures

10.19.1 <contentInstance> resource

The <contentInstance> resource shall contain the following attributes referring to sub-resources.

Table 10.210: Resource description

Attribute Name	Presence in createReq	Presence in updateReq	Presence in response	Description
id	O	N/A	M*	Id suggested by the issuer (in the CREATE request) or provided by the hosting SCL (in the response)
href	NP	N/A	O	This is the URI to be used to directly retrieve the contentInstance. Some contentInstance (i.e. a resource that represents a location retrieved from location server) cannot be addressed explicitly, in which case the attribute is optional.
contentTypes	NP	N/A	M	See table 11.36
contentSize	NP	N/A	M	See table 11.36
creationTime	NP	N/A	M	See table 11.36
lastModifiedTime	NP	N/A	M	See table 11.36
delayTolerance	O	N/A	O	See table 11.36
content	M	N/A	M	See table 11.36

10.19.2 contentInstanceCreate

10.19.2.1 contentInstanceCreateRequestIndication

This procedure adds a <contentInstance> resource to the contentInstances resource. The SCL primitive shall comply with the following.

Table 10.211: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	SCL	SCL	Hosting SCL

Table 10.212: ContentInstanceCreateRequestIndication

SCL Primitive: contentInstanceCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_CREATE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to create a contentInstance resource
targetID	M	The URI of the target entity where the contentInstance resource shall be created
Resource	**Mandatory/Optional**	**Description**
contentInstance	O	The contentInstance resource representation Only one of contentInstance or 'raw content' shall be present
raw content	O	raw content [see resource attributes; Content] Only one of contentInstance or 'raw content' shall be present

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID". However, this case the accessRightID of the parent resource (the container resource) shall be used to check the authorization

4) "Check the syntax of received message".

5) Primitive specific operation: If neither 'raw content' nor a contentInstance resource representation are present in the request the request shall be rejected with a STATUS_BAD_REQUEST.

6) Primitive specific operation: if raw content is provided in the request primitive, then the hosting SCL shall construct a *contentInstance* resource representation using the raw content. The delayTolerance shall be absent from the resource representation in this case. The constructed resource representation is used in the rest of the steps.

7) "Check validity of resource representation for create".

8) Primitive specific operation: Validate the provided attributes. If the parent of the addressed contentInstances resource is locationContainer resource with the value of the *locationContainerType* attribute set to "LOCATION_SERVER_BASED", the request shall be rejected with a STATUS_FORBIDDEN.

9) "Create the resource". In case of "raw content" the hosting SCL shall create the resource representation based on the meta-data information received from the transport layer.

10) Primitive specific operation: If the addition of the resource violates the policies 'maxNrOfInstances", "maxByteSize", the hosting SCL shall remove as many of the oldest instances from the collection as is needed to satisfy the policies.

11) Primitive specific operation: the lastModifiedTime (and possibly the e-tag) of the parent contentInstances collection resource shall be set to the same time as the creationTime of the just created contentInstance resource. Note that the modification of the parent contentInstances collection resource may trigger notifications to be sent on active subscriptions.

12) "Create a successful ResponseConfirm".

13) "Send ResponseConfirm primitive".

Post response actions:

14) Primitive specific operation: If a maxInstanceAge attribute value is positive in the parent resource of the addressed contentInstances resource then, the hosting SCL shall start a timer for this new contentInstance that expires at the limitation of maxInstanceAge. If this timer expires, the hosting SCL shall delete the <contentInstance> as described in "Delete the addressed resource".
The timer shall be stopped when the contentInstance is deleted for any reason, e.g. by hosting SCL due to timer expiry or because the contentInstance resource is deleted by the contentInstanceDelete procedures.

10.19.2.2 contentInstanceCreateResponseConfirm (successful case)

Confirms the creation of a new <contentInstance>. The SCL primitive shall comply with the following.

Table 10.213: ContentInstanceCreateResponseConfirm (successful case)

SCL primitive: contentInstanceCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/optional**	**Description**
contentInstance	O	Full resource representation of the instance. This is only present if any of the provided attributes where modified by the hosting SCL

10.19.2.3 contentInstanceCreateResponseConfirm (unsuccessful case)

The SCL primitive shall comply with the following.

Table 10.214: ContentInstanceCreateResponseConfirm (unsuccessful case)

SCL primitive: contentInstanceCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_CONTANCE_CREATE_RESPONSE
errorInfo	M	Provides error information

10.19.3 contentInstanceRetrieve

10.19.3.1 contentInstanceRetrieveRequestIndication

This request is used to read a <contentInstance> resource. This operation is used to read the entire resource or an individual attribute. The SCL primitive shall comply with the following.

Table 10.215: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.216: ContentInstanceRetrieveRequestIndication

SCL Primitive: contentInstanceRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the target entity where the content instance resource to be addressed. When the grandparent resource of contentInstance is container or a locationContainer resource with the type of "LOCATION_SERVER_BASED", the URI can also point at the latest sub-resource or oldest sub-resource of the contentInstances resource: .../contentInstances/latest .../contentInstances/oldest
primitiveType	M	CONTENT_INSTANCE_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) Primitive specific operation: the hosting SCL shall check if the parent contentInstance resource exists. If the contentInstances resource does not exist, the request shall be rejected with a STATUS_NOT_FOUND.

3) "Check authorization of the requestingEntity based on accessRightID". However, this case the accessRightID of the grand-parent resource (the container resource) shall be used to check the authorization.

4) "Check the syntax of received message".

5) Primitive specific operation: The hosting SCL shall check the grand-parent of the addressed contentInstance resource. If the grandparent is of type locationContainer and the *locationContainerType* attribute of that locationContainer is "LOCATION_SERVER_BASED" then the operation shall continue with step 6. Otherwise, i.e. if the grandparent is a resource of type container or if it is a locationContainer with a *locationContainerType* attribute with a value "APPLICATION_GENERATED" then the sequence shall continue with step 9.

6) **LocationContainer of type LOCATION_SERVER_BASED specific handling**
Primitive specific operation: The hosting SCL shall check the targetID. If the targetID is not …/contentInstances/oldest or …/contentInstances/latest. If so, then the hosting SCL shall reject the issuer with STATUS_FORBIDDEN.

7) The hosting SCL shall retrieve the *locTargetDevice* attribute from the <scl> great-grandparent of the addressed *contentInstance* resource (i.e. the <scl> resource under which the parent locationContainer resource resides)
The hosting SCL shall also retrieve the *locRequestor* attribute from the resource corresponding to the requestingEntity (i.e. an <application> resource).
If the requestingEntity is an SCL, the hosting SCL shall reject the request with a STATUS_FORBIDDEN.
In case either the locTargetDevice or the locRequestor cannot be obtained, the hosting SCL shall reject the request with a STATUS_FORBIDDEN.
Then the hosting SCL shall transform the received RequestIndication into LCS (LoCation Services) request, using the obtained locRequestor and locTargetDevice attributes.
The hosting SCL shall provide default values for other attributes (e.g. location accuracy) required in the LCS service request (e.g. complying with Le interface defined in OMA Mobile Location Protocol) according to local policies.
The hosting SCL shall send this LCS service request to the location server, e.g. through Le interface.

8) Primitive specific operation:
The hosting SCL shall receive the corresponding LCS service response from the location server and transform it into a ResponseConfirm primitive. Note that the location server performs the privacy control and only responds successfully if the positioning procedure is permitted.
If the LCS service response indicates the request was unsuccessful, then the hosting SCL shall reject the RequestIndication. The tables in clause 10.18.3 provides a statusCode mapping for the OMA MLS statusCodes. In case OMA MLS is used, then this mapping shall be applied. The type of error information for items7, 10-20, 24, 30-34 depends on the local policies. If NSCL will revise the LCS service request for the requestingEntity when receiving the future request from this requestingEntity, then the hosting SCL shall respond to the issuer with STATUS_INTERNAL_SERVER_ERROR. Otherwise, if the NSCL will not revise the future request, then the hosting SCL shall respond to the issuer with STATUS_BAD_REQUEST. This completes the retrieval procedure between hosting SCL and location server.
Continue with step 12. Step 9-11 shall be skipped.

9) **Container or locationContainer of type APPLICATION_GENERATED**
Primitive specific operation: The hosting SCL shall check if the targetID addresses ./contentInstances/latest or …/contentInstances/oldest. If so then:

   a) The hosting SCL shall check if the collection of contentInstance resources is empty. If so, the request is rejected with a STATUS_NOT_FOUND.

   b) The hosting SCL shall continue with the remainder of the steps as if the oldest or the latest instance in the collection is addressed in the targetID. The oldest instance is the contentInstance resource in the collection that has the chronologically oldest value for the creationTime attribute, the latest instance is the contentInstance resource in the collection that has the chronologically newest value for the creationTime attribute.

10) "Read addressed resource".

11) "Create a successful ResponseConfirm".

12) "Send ResponseConfirm primitive".

10.19.3.2 contentInstanceRetrieveResponseConfirm (successful case)

This response is triggered by the contentInstanceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.217: ContentInstanceRetrieveResponseConfirm (successful case)

SCL primitive: contentInstancesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
contentInstance	M	Full representation of the retrieved contentInstance resource

10.19.3.3 contentInstanceRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the contentInstanceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.218: ContentInstanceRetrieveResponseConfirm (unsuccessful case)

SCL primitive: contentInstancesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.19.4 contentInstanceUpdate

The contentInstance resource shall not be updated. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.19.5 contentInstanceDelete

10.19.5.1 contentInstanceDeleteRequestIndication

This request is used to delete a <contentInstance> resource. The SCL primitive shall comply with the following.

Table 10.219: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	SCL	SCL	Hosting SCL

Table 10.220: ContentInstanceDeleteRequestIndication

SCL Primitive: contentInstanceDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_DELETE_REQUEST
requestingEntity	M	The entity (an application or SCL) that requests to delete a contentInstance resource
targetID	M	The URI of the target <contentInstance> location

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) Primitive specific operation: the hosting SCL shall check if the parent contentInstance resource exists. If the contentInstances resource does not exist, the request shall be rejected with a STATUS_NOT_FOUND.

3) "Check authorization of the requestingEntity based on accessRightID". However, this case the accessRightID of the grand-parent resource (the container resource) shall be used to check the authorization.

4) "Check the syntax of received message".

5) Primitive specific operation: The hosting SCL shall check if the targetID addresses ../contentInstances/latest or …/contentInstances/oldest. If so then:

   a) The hosting SCL shall check if the collection of contentInstance resources is empty. If so, the request is rejected with a STATUS_NOT_FOUND.

   b) The hosting SCL shall continue with the remainder of the steps as if the oldest or the latest instance in the collection is addressed in the targetID. The oldest instance is the contentInstance resource in the collection that has the chronologically oldest value for the creationTime attribute, the latest instance is the contentInstance resource in the collection that has the chronologically newest value for the creationTime attribute.

6) "Check existence of the addressed resource".

7) "Delete the addressed resource".

8) "DeAnnounce resource".

9) "Create a successful ResponseConfirm".

10) "Send ResponseConfirm primitive".

Post result operations:

11) The hosting SCL shall stop any associated maxInstanceAge timer.

10.19.5.2 contentInstanceDeleteResponseConfirm(successful case)

The SCL primitive shall comply with the following.

Table 10.221: ContentInstanceDeleteResponseConfirm (successful case)

SCL primitive: contentInstanceDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.19.5.3 contentInstanceDeleteResponseConfirm (unsuccessful case)

This response is issued if the contentInstanceDeleteRequestIndication was not successfully serviced. The SCL primitive shall comply with the following.

Table 10.222: ContentInstanceDeleteResponseConfirm (unsuccessful case)

SCL primitive: contentInstanceDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	CONTENT_INSTANCE_DELETE_RESPONSE
errorInfo	M	Provides error information

10.20 groups resource and management procedures

10.20.1 groups resource

The *groups* resource representation shall contain the following sub-resources and attributes.

Table 10.223: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
groupCollection	N/A	NP	M	See table 11.37
groupAnncCollection	N/A	NP	M	See table 11.37
subscriptionsReference	N/A	NP	M	See table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.20.2 groupsCreate

The groups resource shall be created as part of the parent and it shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.20.3 groupsRetrieve

10.20.3.1 groupsRetrieveRequestIndication

This request is used for retrieving the content of a groups resource. The SCL primitive shall comply with the following.

Table 10.224: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.225: GroupsRetrieveRequestIndication

SCL Primitive: groupsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the groups resource to be addressed
primitiveType	M	GROUPS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.20.3.2 groupsRetrieveResponseConfirm (successful case)

This response is triggered by the groupsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.226: GroupsRetrieveResponseConfirm (successful case)

SCL primitive: groupsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUPS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
groups	M	Full representation of the retrieved groups resource

10.20.3.3 groupsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the groupsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.227: GroupsRetrieveResponseConfirm (unsuccessful case)

SCL primitive: groupsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUPS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.20.4 groupsUpdate

10.20.4.1 groupsUpdateRequestIndication

This request is used to update the full representation of the groups.

The SCL primitive shall comply with the following.

Table 10.228: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.229: GroupsUpdateRequestIndication

SCL Primitive: groupsUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the groups resource to be addressed
primitiveType	M	GROUPS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
groups	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for update".

6) "Update the addressed resource".

7) "Announce resource".

8) "Create a successful response Confirm".

9) "Send response Confirm primitive".

10.20.4.2 groupsUpdateResponseConfirm (successful case)

This response is triggered by the groupsUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.230: GroupsUpdateResponseConfirm (successful case)

SCL primitive: groupsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUPS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
groups	O	Full representation of the updated groups resource, if any of the provided attributes were modified by the hosting SCL.

10.20.4.3 groupsUpdateResponseConfirm (unsuccessful case)

This response is triggered by the groupsUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.231: GroupsUpdateResponseConfirm (unsuccessful case)

SCL primitive: groupsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUPS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.20.5 groupsDelete

The groups resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.21 <group> resource and management procedures

10.21.1 <group> resource

The <group> resource shall contain the following sub-resources and attributes.

Table 10.232: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
membersContentReference	NP	NP	M	See table 11.37
subscriptionsReference	NP	NP	M	See table 11.37
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
announceTo	O	O	M*	See table 11.36
memberType	M	NP	M	See table 11.36
currentNrOfMembers	NP	NP	M	See table 11.36
maxNrOfMembers	O	O	M	See table 11.36
members	O	O	M	See table 11.36
id	O	NP	M*	The ID of the <group> resource. This is used to identify the resource in its parents collection
membersContentAccessRightID	O	O	O	ID of an accessRight resource manages the access right to access the membersContent resource
consistencyStrategy	O	O	M	The attribute determines how to deal with the <group> resource if the memberType validation fails.

10.21.2 groupCreate

10.21.2.1 groupCreateRequestIndication

This primitive creates a new <group> resource in a groups collection. The SCL primitive shall comply with the following.

Table 10.233: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.234: GroupCreateRequestIndication

SCL Primitive: groupCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_CREATE_REQUEST
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created This URI shall be one of the following: <sclBase>/groups <sclBase>/scls/<scl>/groups <sclBase>/applications/<app>/groups <sclBase>/scls/<scl>/applications/<appAnnc>/groups
Resource	**Mandatory/Optional**	**Description**
group	M	The group resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

6) Primitive specific operation: Validate the provided attributes. It shall also check whether the number of URIs present in the *members* attribute of the group resource representation does not exceed the maximum as specified by the attribute *maxNrOfMembers*. If the maximum is exceeded, the request shall be rejected with a STATUS_FORBIDDEN.
The hosting SCL shall verify that any member URI in the attribute *members* of group resource representation provided in the request shall not refer to a remote <group> resource (hosted by a remote SCL). Otherwise, the request shall be rejected with a STATUS_FORBIDDEN. If the *memberType* attribute of the <group> resource is not "MIXED", the hosting SCL shall also verify that all the member URIs in the attribute *members* of the group resource representation provided in the request shall conform to the *memberType* of the group resource by examine each member's URI.

   Upon unsuccessful validation, the receiver shall delete the *<group>* resource if the *consistencyStrategy* of the *<group>* resource is ABANDON_GROUP, or remove the inconsistent members from the *<group>* resource if the *consistencyStrategy* attribute is ABANDON_MEMBER, or set the *memberType* attribute of the *<group>* resource to "MIXED" if the *consistencyStrategy* attribute is MODIFY_TYPE.

7) "Create resource".

8) "Announce resource".

9) "Create a successful ResponseConfirm".

10.21.2.2 groupCreateResponseConfirm (successful case)

Confirms the creation of a new group resource in a groups collection. The SCL primitive shall comply with the following.

Table 10.235: GroupCreateResponseConfirm (successful case)

SCL primitive: groupCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/optional**	**Description**
group	O	Full resource representation of the group. This is only present if any of the provided attributes where modified by the hosting SCL

10.21.2.3 groupCreateResponseConfirm (unsuccessful case)

This response is triggered by the groupCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.236: GroupCreateResponseConfirm (unsuccessful case)

SCL primitive: groupCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_CREATE_RESPONSE
errorInfo	M	Provides error information

10.21.3 groupRetrieve

10.21.3.1 groupRetrieveRequestIndication

This request is used for retrieving the content of a group resource. The SCL primitive shall comply with the following.

Table 10.237: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.238: GroupRetrieveRequestIndication

SCL Primitive: groupRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the group resource to be addressed
primitiveType	M	GROUP_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.21.3.2 groupRetrieveResponseConfirm (successful case)

This response is triggered by the groupRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.239: GroupRetrieveResponseConfirm (successful case)

SCL primitive: groupRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
group	M	Full representation of the retrieved group resource

10.21.3.3 groupRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the groupRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.240: GroupRetrieveResponseConfirm (unsuccessful case)

SCL primitive: groupRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.21.4 groupUpdate

10.21.4.1 groupUpdateRequestIndication

This request is used to update the full representation of the group.

It is also possible to update only part of the group representation, see clause 10.39.

The SCL primitive shall comply with the following.

Table 10.241: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.242: groupUpdateRequestIndication

SCL Primitive: groupUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the group resource to be addressed
primitiveType	M	GROUP_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
group	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for update".

6) Primitive specific operation: Validate the provided attributes. The hosting SCL shall verify that any provided member URI in the attribute *members* of group resource provided in the request shall not refer to a remote <group> resource (hosted by a remote SCL). Otherwise, the request shall be rejected with a STATUS_FORBIDDEN.
   Moreover, if a local <*group*> resource is provided in the attribute *members* as the potential member of the updated group resource, the hosting SCL shall detect the member and descendent members of the local <group> resource to ensure that the existing <*group*> resource shall not be contained in the local <*group*> resource. If the local <*group*> resource includes the existing <*group*> resource as a member or descendent member, the request shall be rejected with a STATUS_FORBIDDEN. This essentially prevents loops from being created.
   If the *memberType* attribute of the <group> resource is not "MIXED", the hosting SCL shall also verify that all the member URIs in the attribute *members* of the group resource representation provided in the request shall conform to the *memberType* of the group resource by examine each member's URI.

   Upon unsuccessful validation, the receiver shall delete the *<group>* resource if the *consistencyStrategy* of the *<group>* resource is ABANDON_GROUP, or remove the inconsistent members from the *<group>* resource if the *consistencyStrategy* attribute is ABANDON_MEMBER, or set the *memberType* attribute of the *<group>* resource to "MIXED" if the *consistencyStrategy* attribute is MODIFY_TYPE.

7) Primitive specific operation: The hosting SCL shall check whether the number of provided *members* in the attribute members exceeds the limitation of maxNrOfMembers. If it exceeds, the hosting SCL shall reject the request with STATUS_FORBIDDEN.

8) "Update the addressed resource".

9) "Announce resource".

10) "Create a successful ResponseConfirm".

11) "Send ResponseConfirm primitive".

10.21.4.2 groupUpdateResponseConfirm (successful case)

This response is triggered by the groupUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.243: GroupUpdateResponseConfirm (successful case)

SCL primitive: groupUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
group	O	Full representation of the updated group resource, if any of the provided attributes were modified by the hosting SCL

10.21.4.3 GroupUpdateResponseConfirm (unsuccessful case)

This response is triggered by the groupUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.244: GroupUpdateResponseConfirm (unsuccessful case)

SCL primitive: groupUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.21.5 groupDelete

10.21.5.1 groupDeleteRequestIndication

The procedure is used to delete a group resource. The SCL primitive shall comply with the following.

Table 10.245: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.246: GroupDeleteRequestIndication

SCL Primitive: groupDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the group resource deletion
targetID	M	The URI of the group resource to be addressed
primitiveType	M	GROUP_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "De-announce resource".

6) "Delete the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.21.5.2 groupDeleteResponseConfirm (successful case)

This response is triggered by the groupDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.247: GroupDeleteResponseConfirm (successful case)

SCL primitive: groupDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.21.5.3 groupDeleteResponseConfirm (unsuccessful case)

This response is triggered by the groupDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.248: GroupDeleteResponseConfirm (unsuccessful case)

SCL primitive: groupDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_DELETE_RESPONSE
errorInfo	M	Provides error information

10.22 <groupAnnc> resource and management procedures

10.22.1 <groupAnnc> resource

The <groupAnnc> resource shall contain the following attributes.

Table 10.249: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
link	M	NP	M	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	M	M	M	See table 11.36
expirationTime	O	O	M*	See table 11.36
id	O	NP	M*	The ID of the announcement resource. This is used to identify the resource in its parents collection. The id is the id of the corresponding <group> resource postfixed with "Annc"

10.22.2 groupAnncCreate

10.22.2.1 groupAnncCreateRequestIndication

This request is used to create a new <groupAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.250: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.251: GroupAnncCreateRequestIndication

SCL Primitive: groupAnncCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created
primitiveType	M	GROUP_ANNC_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
groupAnnc	O	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Create announced resource".

10.22.2.2 groupAnncCreateResponseConfirm (successful case)

It confirms the creation of a new <groupAnnc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.252: GroupAnncCreateResponseConfirm (successful case)

SCL primitive: groupAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
groupAnnc	O	In the case that any of the provided attributes in the request have been modified by the hosting SCL then the complete content of the resource as described above is returned in the response as well

10.22.2.3 groupAnncCreateResponseConfirm (unsuccessful case)

This response is triggered by the groupAnncCreateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.253: GroupAnncCreateResponseConfirm (unsuccessful case)

SCL primitive: groupAnncCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.22.3 groupAnncRetrieve

10.22.3.1 groupAnncRetrieveRequestIndication

This request is used for retrieving the content of an <groupAnnc> resource. The SCL primitive shall comply with the following.

Table 10.254: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.255: GroupAnncRetrieveRequestIndication

SCL Primitive: groupAnncRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the announced resource to be addressed.
primitiveType	M	GROUP_ANNC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Retrieve announced resource".

10.22.3.2 groupAnncRetrieveResponseConfirm (successful case)

This response is triggered by the groupAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.256: GroupAnncRetrieveResponseConfirm (successful case)

SCL primitive: groupAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
groupAnnc	M	Complete content of the resource is returned in the response

10.22.3.3 groupAnncRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the groupAnncRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.257: GroupAnncRetrieveResponseConfirm (unsuccessful case)

SCL primitive: groupAnncRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.22.4 groupAnncUpdate

10.22.4.1 groupAnncUpdateRequestIndication

This request is used to update and to modify the <groupAnnc> resource. The SCL primitive shall comply with the following.

Table 10.258: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.259: GroupAnncUpdateRequestIndication

SCL Primitive: groupAnncUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	GROUP_ANNC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
groupAnnc	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Update announced resource".

10.22.4.2 groupAnncUpdateResponseConfirm (successful case)

This response is triggered by the groupAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.260: GroupAnncUpdateResponseConfirm (successful case)

SCL primitive: groupAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
groupAnnc	O	Full representation of the retrieved announced resource

10.22.4.3 groupAnncUpdateResponseConfirm (unsuccessful case)

This response is triggered by the groupAnncUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.261: GroupAnncUpdateResponseConfirm (unsuccessful case)

SCL primitive: groupAnncUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.22.5 groupAnncDelete

10.22.5.1 groupAnncDeleteRequestIndication

The procedure is used to delete an <groupAnnc> resource. The SCL primitive shall comply with the following.

Table 10.262: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.263: GroupAnncDeleteRequestIndication

SCL Primitive: groupAnncDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the announced resource deletion
targetID	M	The URI of the announced resource to be addressed
primitiveType	M	GROUP_ANNC_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the common procedure "Delete announced resource".

10.22.5.2 groupAnncDeleteResponseConfirm (successful case)

This response is triggered by the groupAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.264: GroupAnncDeleteResponseConfirm (successful case)

SCL primitive: groupAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.22.5.3 groupAnncDeleteResponseConfirm (unsuccessful case)

This response is triggered by the groupAnncDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.265: GroupAnncDeleteResponseConfirm (unsuccessful case)

SCL primitive: groupAnncDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	GROUP_ANNC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.23 membersContent resource and management procedures

10.23.1 membersContent resource

This is a virtual resource.

The membersContent resource is used to fan-out requests to group members and aggregate their responses.

The responses contain a collection of results, one for each of the member URIs in the *members* collection of the group. The aggregated result of a request to the membersContent resource is represented as a non-addressable resource membersContentResponses.

Table 10.266: MembersContentResponses entity description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
membersContentResponses	N/A	N/A	M	

10.23.2 membersContentCreate

10.23.2.1 membersContentCreateRequestIndication

The primitives creates the content of all member resources belonging to an existing group resource. The group hosting SCL primitive shall comply with the following.

Table 10.267: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Group Hosting SCL

Table 10.268: MembersContentCreateRequestIndication

SCL Primitive: membersContentCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_CREATE_REQUEST
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the membersContent resource of the group resource or a subordinate resource
Resource	**Mandatory/Optional**	**Description**
Resource	M	Resource to be created, which is transparent to group hosting SCL

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The targetID consists of the URI of the group resource plus a suffix marked by /membersContent or /membersContent/.....
The group hosting SCL shall derive the URI of the group from the primitive attribute *targetID* and then obtain the *accessRightID* of the group resource.

4) "Check authorization of the requestingEntity based on accessRightID" where the *membersContentAccessRightID* of the parent group resource is used. In the case the *membersContentAccessRightID* is not provided, the *accessRightID* of the parent group resource shall be used."

5) "Check the syntax of received message".

6) "Check validity of resource representation".

7) Primitive specific operation: Validate the provided attributes. If the *memberType* attribute of the addressed parent resource is not "MIXED", the group hosting SCL may check whether the type of *Resource* to be created is consistent with the addressed parent resource. I.e. if .the targetID was …/membersContent without any suffix, then the memberType attribute of the parent group resource determines the type of the addressed resource. Otherwise it is determined by the combination of the memberType and the child resources addressed in the targetID after the membersContent element in the path. If they are not consistent, the request shall be rejected with a STATUS_BAD_REQUEST.

8) Primitive specific operation: The group hosting SCL shall obtain URIs of addressed resources from the attribute *members* of the parent <group> resource. The group hosting SCL may determine that multiple member resources belong to the same remote member hosting SCL, and may perform as an issuer to request to create a sub-group containing the specific multiple member resources in that member hosting SCL. This sub-group is created in the member hosting SCL as described in clause 10.21.2. The targetID of this groupCreate request may be *<memberHosting sclBase>/scls/<groupHostingScl>/groups* or *<memberHosting sclBase>/groups*, etc. The group hosting SCL shall also provide *requestingEntity* (i.e. group hosting SCL) and sub-group resource representation that contains a member attribute with all the members residing on the addressed member Hosting SCL. The sub-group representation may include the attribute *accessRightID,* so that the group hosing SCL has the access right to this sub-group. The id of the sub-group may be proposed by the group hosting SCL and determined by the member hosting SCL or it may be given by the member hosting SCL.
If there is already a sub-group resource defined in the remote member hosting SCL, then the group hosting SCL may utilize the existing sub-group resource.

9) Primitive specific operation: For each member hosting SCL, the group hosting SCL shall perform the following steps:

    a) Primitive specific operation: The primitive attributes including primitiveType, requestingEntity and targetID shall be mapped to the primitive attributes of the corresponding RequestIndication. The primitive attribute primitiveType shall be modified to corresponding primitiveType according to the type of Resource provided in the membersContentCreateRequestIndication. The primitive attribute requestingEntity shall be directly used. The prefix of primitive attribute targetID i.e. <URI of group resource>/membersContent shall be replaced by each URI of members resources derived from the attribute members of the group resource, but excluding the members resources which construct a sub-group. For these members resources contained in a sub-group, the primitive targetID of the composed RequestIndication shall be <URI of sub-group resource>/membersContent. The group hosting SCL shall execute "Compose RequestIndication primitives".

    b) "Send the RequestIndication to the receiver SCL".

    c) "Wait for ResponseConfirm primitives".

10) The procedures between group hosting SCL and member hosting SCLs shall comply with the corresponding creation procedures as described in clause 10. The detailed procedures are according to the type of Resource provided in the membersContentCreateRequestIndication.

11) The group hosting SCL shall unpack the responses, i.e. it shall obtain each member resources responses from the response of the members resources and the sub-groups created by the group hosting SCL and aggregate these into a membersContentResponses:

   a) If the response belongs to a sub-group created by the group hosting SCL, the group hosting SCL shall collect the individual responses from the sub-group membersContentResponses included in the response received from member hosting SCL and add each of these responses to the aggregated membersContentResponses.

   b) If the response belongs to a sub-group not created by the group hosting SCL, the group hosting SCL shall obtain the statusCode and other attributes received in the response of the member hosting SCL and add them into a membersContentResponse created by a group hosting SCL. And the group hosting SCL shall also obtain membersContentResponses from the response of the member hosting SCL and add it into the membersContentResponse created by a group hosting SCL as the resultBody. Then this membersContentResponse shall be added to the aggregated membersContentResponses of the group hosting SCL.

   c) If the response does not belong to a sub-group, then the group hosting SCL shall obtain the statusCode and ,if present, the URI of created member given by a Location header field received from the member hosting SCL and may obtain attributes including etag from the successful response from member hosting SCL. These will be added to the aggregated membersContentResponses.

12) "Create a successful ResponseConfirm" which includes the aggregated membersContentResponses.

13) "Send response Confirm primitive".

10.23.2.2 membersContentCreateResponseConfirm (successful case)

This response is triggered by the membersContentCreateRequestIndication. This confirms the creation of members content in a group resource. The SCL primitive shall comply with the following.

Table 10.269: MembersContentCreateResponseConfirm (successful case)

SCL primitive: membersContentCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_CREATE_RESPONSE
statusCode	M	Provides the status code for the treatment in the group hosting SCL
Resource	**Mandatory/Optional**	**Description**
membersContentResponses	M	A non-addressable resource to aggregate the result of a request to the membersContent resource

10.23.2.3 membersContentCreateResponseConfirm (unsuccessful case)

This response is triggered by the membersContentCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.270: MembersContentCreateResponseConfirm (unsuccessful case)

SCL primitive: membersContentCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_CREATE_RESPONSE
errorInfo	M	Provides error information for the treatment in the group hosting SCL

10.23.3 membersContentRetrieve

10.23.3.1 membersContentRetrieveRequestIndication

This request is used for retrieving the members content of a group resource. The SCL primitive shall comply with the following.

Table 10.271: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Group Hosting SCL

Table 10.272: MembersContentRetrieveRequestIndication

SCL Primitive: membersContentRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	Indicates the membersContent resource of the group resource
primitiveType	M	MEMBERS_CONTENT_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The targetID consists of the URI of the group plus the suffix marked by membersContent or /membersContent/..... The group hosting SCL shall derive the URI of the group from the primitive attribute targetID and then obtain the accessRightID of the group resource.

4) "Check authorization of the requestingEntity based on accessRightID" where the *membersContentAccessRightID* of the parent group resource is used. In the case the *membersContentAccessRightID* is not provided, the *accessRightID* of the parent group resource shall be used.

5) "Check the syntax of received message".

6) "Check validity of resource representation".

7) Primitive specific operation: The group hosting SCL shall obtain URIs of addressed resources from the attribute members of the parent <group> resource. The group hosting SCL may determine that multiple member resources belong to the same remote member hosting SCL, and may perform as an issuer to request to create a sub-group containing the specific multiple member resources in that member hosting SCL. This sub-group is created in the member hosting SCL as described in [See groupCreate]. The targetID of this groupCreate request may be <memberHosting sclBase>/scls/<groupHostingScl>/groups or *<memberHosting sclBase>/groups*, etc. The group hosting SCL shall also provide requestingEntity (i.e. group hosting SCL) and sub-group resource representation that contains a member attribute with all the members residing on the addressed member Hosting SCL. The sub-group representation may include the attribute accessRightID, so that the group hosting SCL has the access right to this sub-group. The id of the sub-group may be proposed by the group hosting SCL and determined by the member hosting SCL or it may be given by the member hosting SCL.

If there is already a sub-group resource defined in the remote member hosting SCL, then the group hosting SCL may utilize the existing sub-group resource.

8) Primitive specific operation: For each member hosting SCL, the group hosting SCL shall perform the following steps:

    a) Primitive specific operation: The primitive attributes including primitiveType, requestingEntity and targetID shall be mapped to the primitive attributes of the corresponding RequestIndication. The primitive attribute primitiveType shall be modified to corresponding primitiveType according to the type of Resource provided in the membersContentCreateRequestIndication. The primitive attribute requestingEntity shall be directly used. The prefix of primitive attribute targetID i.e. <URI of group resource>/membersContent shall be replaced by each URI of members resources derived from the attribute members of the group resource, but excluding the members resources which construct a sub-group. For these members resources contained in a sub-group, the primitive targetID of the composed RequestIndication shall be <URI of sub-group resource>/membersContent.

    b) "Compose RequestIndication primitives", based on the information stated in the previous step

    c) "Send the RequestIndication to the receiver SCL".

    d) "Wait for ResponseConfirm primitives".

    The procedures between group hosting SCL and member hosting SCLs shall comply with the corresponding creation procedures as described in [See SCL Primitives]. The detailed procedures are according to the type of Resource provided in the membersContentCreateRequestIndication.

9) The group hosting SCL shall aggregate the received responses. It shall unpack the responses, ,i.e. obtain each member resources responses from the response of the members r*esources and the sub-groups created* by the group hosting SCL:

    a) If the response belongs to a sub-group created by the group hosting SCL, the group hosting SCL shall collect the individual responses from the sub-group membersContentResponses included in the response received from member hosting SCL and add each of these responses to the aggregated membersContentResponses.

    b) If the response belongs to a sub-group not created by the group hosting SCL, the group hosting SCL shall obtain the statusCode and other attributes received in the response of the member hosting SCL and add them into a membersContentResponse created by a group hosting SCL. And the group hosting SCL shall also obtain membersContentResponses from the response of the member hosting SCL and add it into the membersContentResponse created by a group hosting SCL as the resultBody. Then this membersContentResponse shall be added to the aggregated membersContentResponses of the group hosting SCL.

    c) If the response does not belong to a sub-group, then the group hosting SCL shall obtain the statusCode and ,if present, the resource representation from the response from member hosting SCL. These will be added to the aggregated membersContentResponses.

10) "Create a successful ResponseConfirm" using the aggregated membersContentResponses.

11) "Send response Confirm primitive".

10.23.3.2 membersContentRetrieveResponseConfirm (successful case)

This response is triggered by the membersContentRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.273: MembersContentRetrieveResponseConfirm (successful case)

SCL primitive: membersContentRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_RETRIEVE_RESPONSE
statusCode	M	Provides statusCode for the treatment in the group hosting SCL
Resource	**Mandatory/Optional**	**Description**
membersContentResponses	M	A non-addressable resource to aggregate the result of a request to the membersContent resource

10.23.3.3 membersContentRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the membersContentRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.274: MembersContentRetrieveResponseConfirm (unsuccessful case)

SCL primitive: membersContentRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_RETRIEVE_RESPONSE
errorInfo	M	Provides error information for the treatment in the group hosting SCL

10.23.4 membersContentUpdate

10.23.4.1 membersContentUpdateRequestIndication

This request is used to update the full representation of the member resources.

It is also possible to update only part of the members' representation, see clause 10.39.

The SCL primitive shall comply with the following.

Table 10.275: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Group Hosting SCL

Table 10.276: MembersContentUpdateRequestIndication

SCL Primitive: membersContentUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	Indicates the membersContent resource of the group resource
primitiveType	M	MEMBERS_CONTENT_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
resource	M	The resource that corresponds to the memberType of the group and/or the addressed sub-resource or attribute in the targetID

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The targetID consists of the URI of the group plus the suffix marked by /membersContent or /membersContent/..... The group hosting SCL shall derive the URI of the group from the primitive attribute targetID and then obtain the accessRightID of the group resource.

4) "Check authorization of the requestingEntity based on accessRightID of the group resource" where the *membersContentAccessRightID* of the parent group resource is used. In the case the *membersContentAccessRightID* is not provided, the *accessRightID* of the parent group resource shall be used.

5) "Check the syntax of received message".

6) If the *memberType* attribute of the addressed parent resource is not "MIXED", the group hosting SCL may "Check validity of the resource representation for update" based on the type of the addressed resource. I.e. if the targetID was …/membersContent without any suffix, then the memberType attribute of the parent group resource determines the type of the addressed resource. Otherwise it is determined by the combination of the memberType and the child resources addressed in the targetID after the membersContent element in the path. If it is not consistent the request is rejected with a STATUS_BAD_REQUEST.

7) Primitive specific operation: The group hosting SCL shall obtain URIs of addressed resources from the attribute members of the parent <group> resource. The group hosting SCL may determine that multiple member resources belong to the same remote member hosting SCL, and may perform as an issuer to request to create a sub-group containing the specific multiple member resources in that member hosting SCL. This sub-group is created in the member hosting SCL as described in [See groupCreate]. The targetID of this groupCreate request may be <memberHosting sclBase>/scls/<groupHostingScl>/groups or *<memberHosting sclBase>/groups,* etc. The group hosting SCL shall also provide requestingEntity (i.e. group hosting SCL) and sub-group resource representation that contains a member attribute with all the members residing on the addressed member Hosting SCL. The sub-group representation may include the attribute accessRightID, so that the group hosting SCL has the access right to this sub-group. The id of the sub-group may be proposed by the group hosting SCL and determined by the member hosting SCL or it may be given by the member hosting SCL.
   If there is already a sub-group resource defined in the remote member hosting SCL, then the group hosting SCL may utilize the existing sub-group resource.

8) Primitive specific operation: For each member hosting SCL, the group hosting SCL shall perform the following steps:

   a) Primitive specific operation: The primitive attributes including primitiveType, requestingEntity and targetID shall be mapped to the primitive attributes of the corresponding RequestIndication. The primitive attribute primitiveType shall be modified to corresponding primitiveType according to the type of Resource provided in the membersContentCreateRequestIndication. The primitive attribute requestingEntity shall be directly used. The prefix of primitive attribute targetID i.e. <URI of group resource>/membersContent shall be replaced by each URI of members resources derived from the attribute members of the group resource, but excluding the members resources which construct a sub-group. For these members resources contained in a sub-group, the primitive targetID of the composed RequestIndication shall be <URI of sub-group resource>/membersContent. The group hosting SCL shall execute "Compose RequestIndication primitives".

   b) "Send the RequestIndication to the receiver SCL".

   c) "Wait for ResponseConfirm primitives".

   The procedures between group hosting SCL and member hosting SCLs shall comply with the corresponding update procedures as described in [See groupCreate]. The detailed procedures are according to the type of Resource provided in the membersContentCreateRequestIndication.

9) The group hosting SCL shall aggregate the received responses. For this it needs to unpack the responses, i.e. obtain each member resources responses from the response of the members resource and the sub-groups created by the group hosting SCL:

   a) If the response belongs to a sub-group created by the group hosting SCL, the group hosting SCL shall collect the individual responses from the sub-group membersContentResponses included in the response received from member hosting SCL and add each of these responses to the aggregated membersContentResponses.

   b) If the response belongs to a sub-group not created by the group hosting SCL, the group hosting SCL shall obtain the statusCode and other attributes received in the response of the member hosting SCL and add them into a membersContentResponse created by a group hosting SCL. And the group hosting SCL shall also obtain membersContentResponses from the response of the member hosting SCL and add it into the membersContentResponse created by a group hosting SCL as the resultBody. Then this membersContentResponse shall be added to the aggregated membersContentResponses of the group hosting SCL.

   c) If the response does not belong to a sub-group, then the group hosting SCL shall obtain the statusCode from the member hosting SCL and may obtain attributes including etag and resource representation from the successful response from member hosting SCL. These will be added to the aggregated membersContentResponses.

10) "Create a successful ResponseConfirm".

11) "Send response Confirm primitive".

10.23.4.2 membersContentUpdateResponseConfirm (successful case)

This response is triggered by the membersContentUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.277: MembersContentUpdateResponseConfirm (successful case)

SCL primitive: membersContentUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_UPDATE_RESPONSE
statusCode	M	Provides statusCode for the treatment in the group hosting SCL
Resource	**Mandatory/Optional**	**Description**
membersContentResponses	M	A non-addressable resource to aggregate the result of a request of each member resource

10.23.4.3 memberContentUpdateResponseConfirm (unsuccessful case)

This response is triggered by the membersContentUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.278: MembersContentUpdateResponseConfirm (unsuccessful case)

SCL primitive: membersContentUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_UPDATE_RESPONSE
errorInfo	M	Provides error information for the treatment in the group hosting SCL

10.23.5 membersContentDelete

10.23.5.1 membersContentDeleteRequestIndication

The procedure is used to delete member resources of a group resource. The group hosting SCL primitive shall comply with the following.

Table 10.279: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Group Hosting SCL

Table 10.280: MembersContentDeleteRequestIndication

SCL Primitive: membersContentDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the resource deletion
targetID	M	Indicates the membersContent resource of the group resource
primitiveType	M	MEMBERS_CONTENT_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The targetID consists of the URI of the group plus the suffix marked by /membersContent or /membersContent/..... The group hosting SCL shall derive the URI of the group from the primitive attribute targetID and then obtain the accessRightID of the group resource.

4) "Check authorization of the requestingEntity based on accessRightID of the group resource" where the *membersContentAccessRightID* of the parent group resource is used. In the case the *membersContentAccessRightID* is not provided, the *accessRightID* of the parent group resource shall be used.

5) "Check the syntax of received message".

6) Primitive specific operation: The group hosting SCL shall obtain URIs of addressed resources from the attribute members of the parent <group> resource. The group hosting SCL may determine that multiple member resources belong to the same remote member hosting SCL, and may perform as an issuer to request to create a sub-group containing the specific multiple member resources in that member hosting SCL. This sub-group is created in the member hosting SCL as described in [See groupCreate]. The targetID of this groupCreate request may be <memberHosting sclBase>/scls/<groupHostingScl>/groups or *<memberHosting sclBase>/groups*, etc. The group hosting SCL shall also provide requestingEntity (i.e. group hosting SCL) and sub-group resource representation that contains a member attribute with all the members residing on the addressed member Hosting SCL. The sub-group representation may include the attribute accessRightID, so that the group hosting SCL has the access right to this sub-group. The id of the sub-group may be proposed by the group hosting SCL and determined by the member hosting SCL or it may be given by the member hosting SCL.
If there is already a sub-group resource defined in the remote member hosting SCL, then the group hosting SCL may utilize the existing sub-group resource.

7) Primitive specific operation: For each member hosting SCL, the group hosting SCL shall perform the following steps:

   a) Primitive specific operation: The primitive attributes including primitiveType, requestingEntity and targetID shall be mapped to the primitive attributes of the corresponding RequestIndication. The primitive attribute primitiveType shall be modified to corresponding primitiveType according to the type of Resource provided in the membersContentCreateRequestIndication. The primitive attribute requestingEntity shall be directly used. The prefix of primitive attribute targetID i.e. <URI of group resource>/membersContent shall be replaced by each URI of members resources derived from the attribute members of the group resource, but excluding the members resources which construct a sub-group. For these members resources contained in a sub-group, the primitive targetID of the composed RequestIndication shall be <URI of sub-group resource>/membersContent. The group hosting SCL shall execute "Compose RequestIndication primitives".

   b) "Send the RequestIndication to the receiver SCL".

   c) "Wait for ResponseConfirm primitives".

   The procedures between group hosting SCL and member hosting SCLs shall comply with the corresponding delete procedures as described in clause 10.6. The detailed procedures are according to the type of Resource provided in the membersContentCreateRequestIndication.

8) The group hosting SCL shall unpack the successful responses, i.e. obtain each member resources responses from the response of the members resources and the sub-groups created by the group hosting SCL:

   a) If the response belongs to a sub-group created by the group hosting SCL, the group hosting SCL shall collect the individual responses from the sub-group membersContentResponses included in the response received from member hosting SCL and add each of these responses to the aggregated membersContentResponses.

   b) If the response belongs to a sub-group not created by the group hosting SCL, the group hosting SCL shall obtain the statusCode and other attributes received in the response of the member hosting SCL and add them into a membersContentResponse created by a group hosting SCL. And the group hosting SCL shall also obtain membersContentResponses from the response of the member hosting SCL and add it into the membersContentResponse created by a group hosting SCL as the resultBody. Then this membersContentResponse shall be added to the aggregated membersContentResponses of the group hosting SCL.

   c) If the response does not belong to a sub-group, then the group hosting SCL shall obtain the statusCode from the response from member hosting SCL. These will be added to the aggregated membersContentResponses.

9) "Create a successful ResponseConfirm".

10) "Send response Confirm primitive".

10.23.5.2 membersContentDeleteResponseConfirm (successful case)

This response is triggered by the membersContentDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.281: MembersContentDeleteResponseConfirm (successful case)

SCL primitive: membersContentDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_DELETE_RESPONSE
statusCode	M	Provides statusCode for the treatment in the group hosting SCL
Resource	**Mandatory/Optional**	**Description**
membersContent Response	M	A non-addressable resource to aggregate the result for each member resource

10.23.5.3 membersContentDeleteResponseConfirm (unsuccessful case)

This response is triggered by the membersContentDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.282: MembersContentDeleteResponseConfirm (unsuccessful case)

SCL primitive: membersContentDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MEMBERS_CONTENT_DELETE_RESPONSE
errorInfo	M	Provides error information for the treatment in the group hosting SCL

10.24 subscriptions resource and management procedures

10.24.1 subscriptions resource

The subscriptions resource shall contain the following attributes referring to sub-resources.

Table 10.283: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
subscriptionCollection	N/A	NP	M	references to subscription sub-resources

10.24.2 subscriptionsCreate

The subscriptions resource is created as a child resource of a resource that can be subscribed to (i.e. the subscribable resource). It is created whenever the subscribable resource is created. It is deleted when the subscribable resource is deleted. It cannot be updated, since it does not have any updatable attributes.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.24.3 subscriptionsRetrieve

10.24.3.1 subscriptionsRetrieveRequestIndication

This primitive is used to retrieve a subscriptions resource. The SCL primitive shall comply with the following.

Table 10.284: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.285: SubscriptionsRetrieveRequestIndication

SCL Primitive: subscriptionsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Indicates the subscriptions collection resource to be retrieved
primitiveType	M	SUBSCRIPTIONS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" "The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: the hosting SCL shall check either if the requestingEntity has delete permission on the parent resource of this subscriptions resource. If it doesn't have the delete permission, the request shall be rejected with a STATUS_FORBIDDEN response.

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm" with the created collection resource representation.

7) "Send ResponseConfirm primitive".

10.24.3.2 subscriptionsRetrieveResponseConfirm (successful case)

This response confirms the retrieval of a subscriptions resource. The SCL primitive shall comply with the following.

Table 10.286: SubscriptionsRetrieveResponseConfirm (successful case)

SCL primitive: subscriptionsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTIONS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
subscriptions	M	The subscriptions resource representation as indicated in clause 10.24.1

10.24.3.3 subscriptionsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the subscriptionsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.287: SubscriptionsRetrieveResponseConfirm (unsuccessful case)

SCL primitive: subscriptionsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTIONS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.24.4 subscriptionsUpdate

The subscriptions resource shall not be updated via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.24.5 subscriptionsDelete

The subscriptions resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.25 <subscription> resource and management procedures

10.25.1 <subscription> resource

The <*subscription*> resource shall contain the following attributes.

Table 10.288: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
expirationTime	O	O	M*	See table 11.36
minimalTimeBetweenNotifications	O	O	O	See table 11.36
delayTolerance	O	O	O	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
filterCriteria	O	NP	O	See table 11.36
subscriptionType	NP	NP	M	See table 11.36
contact	M	NP	M	See table 11.36
id	O	NP	M*	The ID of the subscription resource. This is used to identify the resource in its parents collection

10.25.2 subscriptionCreate

10.25.2.1 subscriptionCreateRequestIndication

This primitive creates a new <*subscription*> resource in a Service Capability Layer. The subscription applies to the resource that is associated with the parent resource of the *subscriptions* collection resource indicated in the targetID.

The SCL primitive shall comply with the following.

Table 10.289: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.290: SubscriptionCreateRequestIndication

SCL Primitive: subscriptionCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the subscriptions resource in which the requested resource shall be created. This is the child of the resource that is being subscribed-to
primitiveType	M	Indicates SUBSCRIPTION_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
subscription	M	An subscription resource representation as defined in clause 10.25.1

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".
2) "Send the RequestIndication primitive to the receiver SCL".
3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) Primitive specific operation: The receiver SCL shall check if the issuer is an application. If the issuer is an application and the addressed resource does NOT reside in the repository of the receiver SCL, then the receiver SCL shall:
    a) Create a local callback URI, which is hierarchically subordinate to its sclBase URI.
    b) Link this callback URI to the application's contact URI. This can either be done by storing the relation or by encoding the relation in the URI (e.g. encoding the application's contact URI as a query parameter in the callback URI. In both cases, if there is already a callback URI related to the application's contact URI, this same callback URI can be reused.
    c) Replace the value of the *contactURI* in the *subscription* resource in the create request with the callback URI derived in the previous steps.
2) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
3) "Check existence of the addressed resource".
4) Primitive specific operation: The hosting SCL shall check if the requestingEntity has READ permission on the parent resource of the addressed subscriptions collection resource. If it does not have READ permission the request shall be rejected with STATUS_PERMISSION_DENIED.
5) "Check the syntax of received message".
6) "Check validity of resource representation".
7) "Create the resource".
8) The receiver shall execute "Create a successful ResponseConfirm".
9) "Send ResponseConfirm primitive".

Post-result actions:

10) If either:
    a) the value of the ifNoneMatch member in subscription resources' filterCriteria does not correspond to the current e-tag of the subscribed-to resource; or
    b) if the ifModifiedSince attribute from the subscription resources' filterCriteria attribute indicates a date that is the same or earlier than the lastModifiedTime attribute of the subscribed-to resource;

    then:

    - the hosting SCL shall send an initial subscriptionNotifyRequestIndication primitive to the contact as described in clause 10.25.7.
11) The hosting SCL shall start monitoring the parent resource of the addressed resource collection for modification matching the filterCriteria attribute of the subscription resource representation. See clause 10.25.7.

10.25.2.2 subscriptionCreateResponseConfirm (successful case)

This primitive indicates a successful response after the creation of a new <subscription> resource in a Service Capability Layer.

Table 10.291: SubscriptionCreateResponseConfirm (successful case)

SCL Primitive: subscriptionCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	Indicates SUBSCRIPTION_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	Link to the created subscription resource
Resource	**Mandatory/Optional**	**Description**
subscription	O	A full subscription resource representation as defined in clause 10.25.1. Only present if the hosting SCL modified any of the attributes provided by the Issuer

10.25.2.3 subscriptionCreateResponseConfirm (unsuccessful case)

This primitive indicates an unsuccessful response after an attempt to create of a new <subscription> resource in a Service Capability Layer.

Table 10.292: SubscriptionCreateResponseConfirm (unsuccessful case)

SCL Primitive: subscriptionCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_CREATE_RESPONSE
errorInfo	M	Status code of the unsuccessful response

10.25.3 subscriptionRetrieve

10.25.3.1 subscriptionRetrieveRequestIndication

This primitive reads a <subscription> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.293: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.294: SubscriptionRetrieveRequestIndication

SCL Primitive: subscriptionRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieval of the addressed resource
targetID	M	Link to the subscription resource to be read
primitiveType	M	SUBSCRIPTION_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The hosting SCL shall check if either the requestingEntity matches the requestingEntity matches the original creator of the subscription resource or if the requestingEntity has delete permission on the subscribed-to resource (i.e. on the grandparent of the subscription resource). If neither case is valid and if the requestingEntity has discovery permission on the subscribed-to resource, the request shall be rejected with a STATUS_NOT_FOUND. If neither case is valid and if the requestingEntity doesn't have discovery permission on the subscribed-to resource, the request shall be rejected with a STATUS_PERMISSION_DENIED.

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm" with the retrieved resource representation.

7) "Send ResponseConfirm primitive".

10.25.3.2 subscriptionRetrieveResponseConfirm (successful case)

This primitive indicates a successful response on reading an existing <subscription> resource in a Service Capability Layer.

Table 10.295: SubscriptionRetrieveResponseConfirm (successful case)

SCL Primitive: subscriptionRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
subscription	M	An subscription resource full representation as defined in clause 10.25.1

10.25.3.3 subscriptionRetrieveResponseConfirm (unsuccessful case)

This primitive indicates an unsuccessful response after an attempt to read a <subscription> resource in a Service Capability Layer.

Table 10.296: SubscriptionRetrieveResponseConfirm (unsuccessful case)

SCL Primitive: subscriptionRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_RETRIEVE_RESPONSE
errorInfo	M	Status code of the unsuccessful response

10.25.4 subscriptionUpdate

10.25.4.1 subscriptionUpdateRequestIndication

It updates the provided attributes of an existing <subscription> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.297: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.298: SubscriptionUpdateRequestIndication

SCL Primitive: subscriptionUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the resource update
targetID	M	URI of the subscription resource to be updated
primitiveType	M	SUBSCRIPTION_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
subscription	M	A subscription resource representation as defined in clause 10.3.2.16 that replaces the current representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive.

2) "Send RequestIndication primitive to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting".

2) The receiver shall not execute following steps if the request is re-targeted.

3) "Check existence of the addressed resource".

4) Primitive specific operation: The hosting SCL shall check if the requestingEntity matches the original subscriber. If not, the request shall be rejected with a STATUS_PERMISSION_DENIED response. If the requestingEntity has discovery permission on the subscribed-to resource (i.e. the grand-parent resource of the subscription resource), the request shall be rejected with a STATUS_NOT_FOUND.

5) "Check the syntax of received message".

6) "Update the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

Post-result actions:

9) If the expirationTime attribute is modified, the hosting SCL shall re-start the timer corresponding to the end-time as indicated in the expirationTime attribute of the subscription resource. When this timer expires, the resource shall be deleted by the hosting SCL as indicated in clause 10.3.2.16.

10.25.4.2 subscriptionUpdateResponseConfirm (successful case)

This primitive indicates a successful response on updating of an existing <subscription> resource in a Service Capability Layer.

Table 10.299: SubscriptionUpdateResponseConfirm (successful case)

SCL Primitive: subscriptionUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
subscription	O	A <subscription> resource full representation as defined in clause 10.25.1. Only present if the hosting SCL modified any of the attributes provided by the Issuer

10.25.4.3 subscriptionUpdateResponseConfirm (unsuccessful case)

This primitive indicates an unsuccessful response after an attempt to update a <subscription> resource in a Service Capability Layer.

Table 10.300: SubscriptionUpdateResponseConfirm (unsuccessful case)

SCL Primitive: subscriptionUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_UPDATE_RESPONSE
errorInfo	M	Status code of the unsuccessful response

10.25.5 subscriptionDelete

10.25.5.1 subscriptionDeleteRequestIndication

This primitive deletes a <subscription> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.301: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.302: SubscriptionDeleteRequestIndication

SCL Primitive: subscriptionDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the deletion
targetID	M	URI of the resource to be deleted
primitiveType	M	SUBSCRIPTION_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive to receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) Primitive specific operation: The hosting SCL shall check if the requestingEntity is the same entity as the one that created the subscription. If not, the request shall be rejected with a STATUS_PERMISSION_DENIED. The hosting SCL shall check if the requestingEntity has discovery permission on the subscribed-to resource (the grand-parent resource of the subscription resource), the request shall be rejected with a STATUS_NOT_FOUND.

4) "Check the syntax of received message".

5) "Delete the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.25.5.2 subscriptionDeleteResponseConfirm (successful case)

This primitive indicates a successful response on deletion of an existing <subscription> resource in a Service Capability Layer.

Table 10.303: SubscriptionDeleteResponseConfirm (successful case)

SCL Primitive: subscriptionDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.25.5.3 subscriptionDeleteResponseConfirm (unsuccessful case)

This primitive indicates an unsuccessful response after an attempt to delete a <subscription> resource in a Service Capability Layer.

Table 10.304: SubscriptionDeleteResponseConfirm (unsuccessful case)

SCL Primitive: subscriptionCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_DELETE_RESPONSE
errorInfo	M	Status code of the unsuccessful response

10.25.6 Notify representation

A notify representation which is sent as a response to a subscription shall contain similar information to what otherwise would have been present in a retrieve on the subscribed-to resource. The notify request shall contain the following information in the notify representation.

Table 10.305: Notify representation

AttributeName	presence in subscriptionNotifyRequest	Description
statusCode	M	The status code, that would correspond to a retrieve request on the Subscribed-to resource. In case of partial addressing (see clause 10.39) this corresponds to the retrieve using the attribute accessor from the filter-criteria. If the status code indicates an error (i.e. any code except STATUS_OK, STATUS_CREATED or STATUS_ACCEPTED), then this also indicates that the subscription resource has been deleted.
representation	M	The representation of the Subscribed-to resource in case the status code indicated "success". The representation shall be modified to match the filterCriteria is present in the subscription. See [Complex Datatypes; FilterCriteria] for details. In case the statusCode indicates an error, the representation contains additional error information that would have been present in the corresponding retrieve response to the Subscribed-to Resource, i.e. the errorInfo.
subscriptionReference	M	Reference to the <*subscription*> resource that generated this notification.

10.25.7 subscriptionNotify

10.25.7.1 subscriptionNotifyRequestIndication (normal case)

Whenever the resource that is subscribed-to is modified in a way that matches the *filterCriteria* attribute, the hosting SCL shall notify the subscriber at the contact-URI. It uses the following subscriptionNotifyRequestIndication primitive for this.

The definition of what counts as a modification can be found in clause 11.4.

TS 102 690 [2] specifies how long the hosting SCL waits before sending this primitive.

The SCL primitive shall comply with the following.

Table 10.306: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Local SCL	Local SCL	Hosting SCL
Receiver	Application	Application	Local SCL

Table 10.307: subscriptionNotifyRequestIndication

SCL Primitive: subscriptionNotifyRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	The identity of the hosting SCL, i.e. the SCL where the subscription resource located
targetID	M	The contactURI from the subscription
primitiveType	M	SUBSCRIPTION_NOTIFY_REQUEST
Resource	**Mandatory/Optional**	**Description**
notify	M	The notify structure as described above. The representation contains the current resource representation of the subscribed-to resource. In case of partial addressing, the representation only contains the partial representation that corresponds to the attribute accessor in the filterCriteria. The data structure is defined in clause 10.25.6

The **issuer** shall perform the following steps in order:

Primitive specific operations:

a) If the contactURI refers to a local contentInstances resource, the hosting SCL shall:

1) add the notification data structure as a contentInstance to the container. This may result in removal of older contentInstances from the container.

b) If the contactURI refers to a resource on a remote SCL, the hosting SCL shall:

1) Compose RequestIndication primitive".

2) "Send RequestIndication primitive to receiver SCL".

3) "Wait for ResponseConfirm primitive".

4) Perform the actions specified in the subscriptionNotifyResponseConfirm (normal case).

The **receiver** shall perform the following steps in order:

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check the syntax of received message".

3) Primitive specific operation: If the targetID addresses a contentInstances resource, then the hosting SCL shall add the notification data structure as a contentInstance to the addressed container. This may result in removal of older contentInstances from the container.

4) Primitive specific operation: If the targetID addresses a local URI that is related to an applications contact URI, then the hosting SCL shall:

a) retrieve the corresponding application's contact URI;

b) resolve the retrieved contact URI, e.g. using DNS (e.g. on the NSCL) or on the internal network (e.g. on the gateway);

c) route the NOTIFY to the resolved address;

d) wait for the response.

5) "Create a successful ResponseConfirm".

6) "Send ResponseConfirm primitive".

10.25.7.2 subscriptionNotifyResponseConfirm (response to normal case)

This primitives indicates a successful response on reading an existing <subscription> resource in a Service Capability Layer.

Table 10.308: SubscriptionNotifyResponseConfirm (response to normal case)

SCL Primitive: subscriptionNotifyResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_NOTIFY_RESPONSE
statusCode	M	Status code of the response

The **receiver** shall perform the following steps in order:

Primitive specific operations:

a) If a result is received with a "STATUS_INTERNAL_SERVER_ERROR", "STATUS_NOT_IMPLEMENTED", "STATUS_BAD_GATEWAY", "STATUS_SERVICE_UNAVAILABLE" or "STATUS_GATEWAY_TIMEOUT" statusCode the hosting SCL may:

   1) retry sending the notification at a later time; or

   2) do no further processing.

b) If a result is received with another statusCode the hosting SCL shall:

   1) do no further processing.

c) If no response is received, the hosting SCL may:

   1) retry sending the notification at a later time; or

   2) conclude that the contact is not reachable and modify the scl resources accordingly; or

   3) do no further processing.

10.25.7.3 subscriptionNotifyRequestIndication (error case)

The hosting SCL shall delete the <subscription> resource in the following cases:

1) The <*subscription*> resources' expiration timer expires.

2) The parent resource is removed. This happens when the subscribed-to resource is removed.

3) The subscriber/creator no longer has READ access to the subscribed-to resource. (it may be that the hosting SCL only checks the access permissions when the resource is modified).

Primitive specific operations:

In the cases mentioned above the hosting SCL shall:

1) Delete the subscription resource and remove its reference from the subscriptions collection resource.

2) Create a notify data structure with a statusCode attribute set to "STATUS_EXPIRED" statusCode (for case 1), STATUS_DELETED (for case 2) or a STATUS_PERMISSION_DENIED (for case 3).

3) If the contact in the subscription refers to a local container or contentInstances resource, the hosting SCL shall:

   a) add the notification data structure as a contentInstance to the container. This may result in removal of older contentInstances from the container.

4) If the contactURI refers to a resource on a remote SCL, the hosting SCL shall:

   a) Send a subscriptionNotifyRequestIndication according to the table below.

   b) Wait for the result.

   c) Perform the actions specified in the subscriptionNotifyResponseConfirm (response to error case).

The SCL primitive shall comply with the following.

Table 10.309: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Local SCL	Local SCL	Hosting SCL
Receiver	Application	Application	Local SCL

Table 10.310: SubscriptionNotifyRequestIndication

SCL Primitive: subscriptionNotifyRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	The identity of the hosting SCL, i.e. the SCL where the subscription resource located
targetID	M	The contactURI from the subscription
primitiveType	M	SUBSCRIPTION_NOTIFY_REQUEST
Resource	**Mandatory/Optional**	**Description**
notify	M	The notify structure, which includes the reference to the subscription and the statusCode The representation shall include the errorInfo element in this case

10.25.7.4 subscriptionNotifyResponseConfirm (response to error case)

This primitives indicates a successful response on subscriptionNotifyRequestIndication that was send with an error information in the notify.

Table 10.311: SubscriptionNotifyResponseConfirm (response to error case)

SCL Primitive: subscriptionNotifyResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	SUBSCRIPTION_NOTIFY_RESPONSE
statusCode	M	Status code of the response

Primitive specific operations:

a) If a result is received with a "STATUS_INTERNAL_SERVER_ERROR", "STATUS_NOT_IMPLEMENTED", "STATUS_BAD_GATEWAY", "STATUS_SERVICE_UNAVAILABLE" or "STATUS_GATEWAY_TIMEOUT" statusCode the hosting SCL may:

   1) retry sending the notification at a later time; or

   2) do no further processing.

b) If a result is received with another statusCode the hosting SCL shall:

   1) do no further processing.

c) If no response is received, the hosting SCL may:

   1) retry sending the notification at a later time; or

   2) conclude that the contact is offline and modify the scl or application resources accordingly, or do no further processing.

10.26 m2mPoCs resource and management procedures

10.26.1 m2mPoCs resource

The m2mPoCs resource shall contain the following sub-resources and attributes.

Table 10.312: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
m2mPocCollection	N/A	NP	M	See table 11.37
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.26.2 m2mPocsCreate

The *m2mPocs* collection resource shall not be created directly via the API. It is created with default values whenever the parent <*scl*> resource is created.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.26.3 m2mPocsRetrieve

10.26.3.1 m2mPocsRetrieveRequestIndication

This primitive is used to retrieve an scls collection resource. The SCL primitive shall comply with the following.

Table 10.313: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.314: m2mPocsRetrieveRequestIndication

SCL Primitive: m2mPocsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the primitive.
targetID	M	The m2mPocs collection resource to be retrieved
primitiveType	M	M2MPOCS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order.

1) "Compose RequestIndication primitive".

2) "Send RequestIndication primitive to the receiver SCL".

3) "Wait for ResponseConfirm".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) Primitive specific operation; if the requestingEntity is not the SCL corresponding to the parent <*scl*> resource, then the hosting SCL shall reject the request with a STATUS_PERMISSION_DENIED.
4) "Check the syntax of the received message".
5) "Read the addressed resource".
6) "Create a successful ResponseConfirm" with the created collection resource representation.
7) "Send ResponseConfirm primitive".

10.26.3.2 m2mPocsRetrieveResponseConfirm (successful case)

This primitive confirms the retrieval of an m2mPocs resource. The SCL primitive shall comply with the following.

Table 10.315: m2mPocsRetrieveResponseConfirm (successful case)

SCL primitive: m2mPocsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOCS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
m2mPocs	M	The m2mPocs resource representation as indicated in clause 10.26.1

10.26.3.3 m2mPocsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the m2mPocsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.316: m2mPocsRetrieveResponseConfirm (unsuccessful case)

SCL primitive: m2mPocsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOCS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.26.4 m2mPocsUpdate

The m2mPocs collection resource shall not be updated via the API.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.26.5 m2mPocsDelete

The m2mPocs collection resource shall not be deleted via the API. It shall be deleted when its parent resource is deleted.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.27 m2mPoc Resource and Management Procedures

10.27.1 m2mPoc resource

The <m2mPoc> resource shall contain the following attributes referring to sub-resources and attributes.

Table 10.317: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
contactInfo	M	M	M	The contact Information that is used by the NSCL to reach the SCL to whom this m2mPoc resource belongs (i.e. the SCL corresponding to the <scl> resource that is the grant-parent of the <m2mPoc> resource). It may be populated with the contactURI of a notificationChannel , for instance in the case the SCL is server capable but resides behind a NAT function. See table 11.36
expirationTime	O	O	M*	See table 11.36
onlineStatus	O	O	M*	See table 11.36
id	O	NP	M*	The identity of the m2mPoc resource.
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36

10.27.2 m2mPocCreate

10.27.2.1 m2mPocCreateRequestIndication

This request is used to create a new <m2mPoc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.318: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	DSCL or GSCL
Receiver	N/A	N/A	NSCL

Table 10.319: m2mPocCreateRequestIndication

SCL Primitive: m2mPocCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resource shall be created
primitiveType	M	M2MPOC_CREATE_REQUEST
Resource	**Mandatory/optional**	**Description**
m2mPoc	M	Refer to the above table for m2mPoc resource definition

Whenever the DSCL or GSCL establishes or re-establishes a communication channel to the NSCL via some access Network, the DSCL or GSCL shall execute the following steps in order as the **issuer**:

1) "Compose RequestIndication primitive". The issuer shall initialize the *contactInfo* attribute in the *m2mPoc* resource representation to address information that can be resolved by the NSCL. The issuer may also suggest an expirationTime for the information.

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

4) Primitive specific operation: Start an internal timer that is shorter than the *expirationTime* attribute value received in the previous step, if present If the ResponseConfirm did not contain a *m2mPoc* resource representation, then the *expirationTime* suggested by the issuer in step 1 shall be used instead. If neither an expirationTime was suggested, nor received, then the resource is not guarded by any *expirationTime.* The issuer SCL may still execute this and the next step.

5) Whenever this internal timer expires, the issuer shall refresh the m2mPoc information by updating the m2mPoc resource using the procedures described in clause 10.27.4, where it shall extend the *expirationTime* attribute value in the resource.

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

6) "Create the resource".

7) Primitive specific operation: the hosting SCL shall update the *onlineStatus* and *serverCapability* attribute of the <*scl*> resource parent of the addressed *m2mPocs* collection resource to maintain the following invariant.
The *onlineStatus* of the <*scl*> resource is ONLINE if there is at least one *m2mPoc* resource in its collection with an *onlineStatus* set to ONLINE or if there is long polling active for the <*scl*> as indicated in TS 102 690 [2] (Resource <scl>). The *onlineStatus* of the <*scl*> resource is NOT_REACHABLE if all the *m2mPoc* resources in its collection have an *onlineStatus* set to NOT_REACHABLE. The *onlineStatus* of the <*scl*> resource is OFFLINE in all other cases (i.e. if there are no *m2mPoc* resources in the collection or if all m2mPoc resource in the collection have their *onlineStatus* attribute set to OFFLINE). The *serverCapability* attribute of the <*scl*> resource shall be TRUE if there is at least one *m2mPoc* resource in the collection.

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.27.2.2 m2mPocCreateResponseConfirm (successful case)

It confirms the creation of a new <m2mPoc> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.320: m2mPocCreateResponseConfirm (successful case)

SCL primitive: m2mPocCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
m2mPoc	O	In case that any of the provided attributes in the request has been modified by the hosting SCL (e.g. expirationTime), then the complete representation of the resource as described above is returned in the response

10.27.2.3 m2mPocCreateResponseConfirm (unsuccessful case)

This response is triggered by the M2mPocCreateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.321: m2mPocCreateResponseConfirm (unsuccessful case)

SCL primitive: m2mPocCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_CREATE_RESPONSE
errorInfo	M	Provides error information

10.27.3 m2mPocRetrieve

10.27.3.1 m2mPocRetrieveRequestIndication

This request is used for retrieving the content of an m2mPoc resource. The SCL primitive shall comply with the following.

Table 10.322: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	SCL
Receiver	N/A	N/A	Hosting SCL

Table 10.323: m2mPocRetrieveRequestIndication

SCL Primitive: m2mPocRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	The SCL originally requesting the retrieving of the resource
targetID	M	The URI of the m2mPoc resource to be retrieved
primitiveType	M	M2MPOC_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of requestingEntity based on default access rights".
4) "Check the syntax of received message".
5) "Read the addressed resource".
6) "Create a successful ResponseConfirm".
7) "Send ResponseConfirm primitive".

10.27.3.2 m2mPocRetrieveResponseConfirm (successful case)

This response is triggered by the M2mPocRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.324: m2mPocRetrieveResponseConfirm (successful case)

SCL primitive: m2mPocRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
m2mPoc	M	Resource representation is returned in the response

10.27.3.3 m2mPocRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the M2mPocRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.325: m2mPocRetrieveResponseConfirm (unsuccessful case)

SCL primitive: m2mPocRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.27.4 m2mPocUpdate

10.27.4.1 m2mPocUpdateRequestIndication

This request is used to update and to modify the <*m2mPoc*> resource, or one or more <*m2mPoc*> resource's attributes. The SCL primitive shall comply with the following.

Table 10.326: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	DSCL or GSCL
Receiver	N/A	N/A	NSCL

Table 10.327: m2mPocUpdateRequestIndication

SCL Primitive: m2mPocUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the primitive
targetID	M	The URI of the m2mPoc resource to be updated
primitiveType	M	M2MPOC_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
m2mPoc	M	Resource representation

The DSCL or GSCL shall update the *m2mPoc* information when the contact Information changes, when it wants to enable or disable an point of contact or when the active *m2mPoc* resource is about to expire according to the *expirationTime* attribute. The issuer may decide to update only a specific attribute using partial addressing (clause 10.39), which would be respectively, *contactInfo*, *onlineStatus* or *expirationTime*. Alternatively, the issuer may update the complete resource, which may allows the issuer to modify multiple attributes at the same time.

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

4) Primitive specific operation: restart the internal timer with the updated expirationTime.

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Check validity of resource representation for update".

6) "Update the addressed resource".

7) Primitive specific operation: The hosting SCL shall update the grand-parent <*scl*> resource to maintain the following invariant; The onlineStatus of the <*scl*> resource is ONLINE if there is at least one *m2mPoc* resource in its collection with an *onlineStatus* set to ONLINE or if there is long polling active for the <*scl*> as indicated in TS 102 690 [2] Resource <scl>. The *onlineStatus* of the <*scl*> resource is NOT_REACHABLE if all the m2mPoc resources in its collection have an *onlineStatus* set to NOT_REACHABLE. The *onlineStatus* of the <*scl*> resource is OFFLINE in all other cases (i.e. if there are no *m2mPoc* resources in the collection or if all *m2mPoc* resource in the collection have their *onlineStatus* attribute set to OFFLINE). The *serverCapability* attribute of the <*scl*> resource shall be TRUE if there is at least one *m2mPoc* resource in the collection.

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.27.4.2 m2mPocUpdateResponseConfirm (successful case)

This response is triggered by the M2mPocUpdateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.328: m2mPocRetrieveResponseConfirm (successful case)

SCL primitive: m2mPocRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
m2mPoc	O	Full representation of the updated m2mPoc resource, only present in case the hosting SCL modified any of the provided attributes

10.27.4.3 m2mPocRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the M2mPocRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.329: m2mPocRetrieveResponseConfirm (unsuccessful case)

SCL primitive: m2mPocUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.27.5 m2mPocDelete

10.27.5.1 m2mPocDeleteRequestIndication

The procedure is used to delete an <m2mPoc> resource. The SCL primitive shall comply with the following.

Table 10.330: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	DSCL or GSCL
Receiver	N/A	N/A	NSCL

Table 10.331: m2mPocDeleteRequestIndication

SCL Primitive: m2mPocDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the deletion
targetID	M	The URI of the resource to be deleted
primitiveType	M	M2MPOC_DELETE_REQUEST

The DSCL or GSCL shall delete the <m2mPoc> resource before it detaches a point of attachment on an access network.

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

4) Primitive specific operation: stop the internal timer for guarding the refresh of the m2mPoc resource.

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Delete the addressed resource".

6) Primitive specific operation: the hosting SCL shall set the *onlineStatus* and *serverCapability* attributes of the grand-parent <*scl*> resource to maintain the following invariant; The *onlineStatus* of the <*scl*> resource is ONLINE if there is at least one *m2mPoc* resource in its collection with an *onlineStatus* set to ONLINE or if there is long polling active for the <scl> as indicated in TS 102 690 [2] Resource <scl>. The *onlineStatus* of the <*scl*> resource is NOT_REACHABLE if all the *m2mPoc* resources in its collection have an *onlineStatus* set to NOT_REACHABLE. The *onlineStatus* of the <*scl*> resource is OFFLINE in all other cases (i.e. if there are no *m2mPoc* resources in the collection or if all *m2mPoc* resource in the collection have their *onlineStatus* attribute set to OFFLINE). The *serverCapability* attribute of the <*scl*> resource shall be TRUE if there is at least one *m2mPoc* resource in the collection.

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.27.5.2 m2mPocDeleteResponseConfirm (successful case)

This response is triggered by the M2mPocDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.332: m2mPocDeleteResponseConfirm (successful case)

SCL primitive: m2mPocDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.27.5.3 m2mPocDeleteResponseConfirm (unsuccessful case)

This response is triggered by the M2mPocDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.333: m2mPocDeleteResponseConfirm (unsuccessful case)

SCL primitive: m2mPocDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	M2MPOC_DELETE_RESPONSE
errorInfo	M	Provides error information

10.28 mgmtObjs resources and management procedures

10.28.1 mgmtObjs resource

The *mgmtObjs* resource shall contain the following sub-resources and attributes.

Table 10.334: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
mgmtObjCollection	N/A	NP	M	URI of <*mgmtObj*> sub-resources, see table 11.37
mgmtCmdCollection	N/A	NP	M	URI of <*mgmtCmd*> sub-resources, see table 11.37
subscriptionsReference	N/A	NP	M	URI of subscription collection, see table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.28.2 mgmtObjsCreate

The *mgmtObjs* resource shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.28.3 mgmtObjsRetrieve

10.28.3.1 mgmtObjsRetrieveRequestIndication

This request is used to retrieve the content of an *mgmtObjs* resource. The SCL primitive shall comply with the following.

Table 10.335: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.336: mgmtObjsRetrieveRequestIndication

SCL Primitive: mgmtObjsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the *mgmtObjs* resource to be addressed This URI shall be one of the following: <sclBase>/scls/<scl>/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/mgmtObjs
primitiveType	M	MGMTOBJS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of the received message".

5) "Read the addressed resource".

6) "Create a collection resource representation".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.28.3.2 mgmtObjsRetrieveResponseConfirm (successful case)

This response is triggered by the mgmtObjsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.337: mgmtObjsRetrieveResponseConfirm (successful case)

SCL primitive: mgmtObjsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
MgmtObjs	M	List of references(URLs)to all the child resources of the retrieved *mgmtObjs* resource, and all the attributes defined

10.28.3.3 mgmtObjsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the mgmtObjsRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.338: mgmtObjsRetrieveResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.28.4 mgmtObjsUpdate

10.28.4.1 mgmtObjsUpdateRequestIndication

This request is used to retrieve the content of an mgmtObjs resource. The SCL primitive shall comply with the following.

Table 10.339: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.340: mgmtObjsUpdateRequestIndication

SCL Primitive: mgmtObjsUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the *mgmtObjs* resource to be addressed This URI shall be one of the following: <sclBase>/scls/<scl>/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/mgmtObjs
primitiveType	M	MGMTOBJS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
mgmtObjs	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of resource representation for update".

6) "Update the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.28.4.2 mgmtObjsUpdateResponseConfirm (successful case)

This response is triggered by the mgmtObjsUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.341: mgmtObjsUpdateResponseConfirm (successful case)

SCL primitive: mgmtObjsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
mgmtObjs	O	Full representation of the updated mgmtObjs resource

10.28.4.3 mgmtObjsUpdateResponseConfirm (unsuccessful case)

This response is triggered by the MgmtObjsUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.342: mgmtObjsUpdateResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjsUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.28.5 mgmtObjsDelete

The mgmtObjs resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.29 <mgmtObj> resources and management procedures

10.29.1 <mgmtObj> resource

The <*mgmtObj*> resource shall contain the following attributes referring to sub-resources.

Table 10.343: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
parametersCollection	NP	NP	M	References to sub-resources <*parameters*>, see table 11.37
subscriptionsReference	NP	NP	M	References to sub-resources *subscriptions,* see table 11.37
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
description	O	O	O	See table 11.36
moID	M	NP	M	See table 11.36
originalMO	O	NP	M	See table 11.36
<moAttribute>	O	O	O	See table 11.36
id	O	NP	M*	The id of the <mgmtObj>
NOTE: For the <*mgmtObj*> resource instances using generic or external data model, there could be zero or multiple <moAttribute> attributes that are optional in createReq, updateReq and response. For ETSI specific <*mgmtObj*> resource instances, the multiplicity and optionality of the <moAttribute> attribute is described in annex E. A mandatory <moAttribute> shall be provided in the createReq unless a default value has been defined for its absence, otherwise the receiver shall reject the createReq.				

10.29.2 mgmtObjCreate

10.29.2.1 mgmtObjCreateRequestIndication

This primitive creates a new <*mgmtObj*> resource in an mgmtObjs collection. The SCL primitive shall comply with the following.

Table 10.344: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.345: mgmtObjCreateRequestIndication

SCL Primitive: mgmtObjCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_CREATE_REQUEST
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created <sclBase>/scls/<scl>/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/mgmtObjs
Resource	**Mandatory/Optional**	**Description**
mgmtObj	M	The <*mgmtObj*> resource representation. If the issuer is D/GSCL, the "orginalMO" attribute shall be provided in the request

The **issuer** shall execute the following steps in order.

1) Primitive specific operation: If the issuer is a D/GSCL of the remote entity, it shall generate the <mgmtObj> resource representation based on the MO information of the remote entity to be exposed.

2) "Compose RequestIndication primitive".

3) "Send the RequestIndication to the receiver SCL".

4) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

If the issuer is an M2M Network Application, the receiver shall perform the following steps, otherwise skip to step 11.

6) "Identify the managed remote entity and the management protocol".

7) Primitive specific operation: the receiver shall generate the MO to be added to the remote entity based on the <*mgmtObj*> resource representation provided in the request primitive. The receiver may determine the target location on the remote entity where the generated MO shall be added based on the "moID" provided in the request primitive and the protocol specific data model being used. The receiver may also choose to let the remote entity decide the target location where the generated MO shall be added using protocol specific mechanism (e.g. 'Inbox' as defined in [50]).

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". If the receiver receives an error response from the remote entity because the MO to be added already exists on the remote entity, the receiver shall check (by using e.g. OMA-DM Get command or TR069 GetParameterValues/GetParameterAttributes command) if the existing MO is the same as the one to be added, then it shall consider the requested primitive as successfully performed instead of sending an unsuccessful ResponseConfirm primitive; otherwise, it shall reject the request with the statusCode set to "STATUS_CONFLICT" in the ResponseConfirm primitive. The receiver shall also record the location where the MO is added to the remote entity in the successful case.

10) The receiver may repeat step 9 in order to add to the remote entity the MOs that are mapped from the mandatory sub-resources (including any descendents) that are required to be created automatically with the default attribute values as specified in annex E.

11) "Create the resource". The receiver shall also set the "originalMO" attribute of the created <*mgmtObj*> resource as the location of the corresponding MO on the remote entity recorded in step 9 or as provided in the <*parameters*> resource representation in the request primitive.

12) "Create a successful ResponseConfirm". It shall also provide in the response the URI of the new <*mgmtObj*> resource created.

13) "Send Response Confirm primitive".

10.29.2.2 mgmtObjCreateResponseConfirm (successful case)

Confirms the creation of a new <*mgmtObj*> resource in an mgmtObjs collection. The SCL primitive shall comply with the following.

Table 10.346: mgmtObjCreateResponseConfirm (successful case)

SCL primitive: mgmtObjCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/optional**	**Description**
mgmtObj	O	Full resource representation of the <*mgmtObj*>. This is only present if any of the provided attributes are modified by the hosting SCL

10.29.2.3 mgmtObjCreateResponseConfirm (unsuccessful case)

This response is triggered by the MgmtObjCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.347: mgmtObjCreateResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_CREATE_RESPONSE
errorInfo	M	Provides error information

10.29.3 mgmtObjRetrieve

10.29.3.1 mgmtObjRetrieveRequestIndication

This request is used for retrieving the content of an <*mgmtObj*> resource. The SCL primitive shall comply with the following.

Table 10.348: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.349: mgmtObjRetrieveRequestIndication

SCL Primitive: mgmtObjRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the <*mgmtObj*> resource to be retrieved
primitiveType	M	MGMTOBJ_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

If the issuer is a Network Application, the receiver shall perform the following steps, otherwise skip to step 9.

If the addressed resource doesn't contain valid information (e.g. if the resource content is empty or obsolete according to local policy), the receiver shall perform the following steps, otherwise skip to step 10.

6) "Identify the managed remote entity and the management protocol".

7) "Locate the MO information to be managed on the remote entity".

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". The receiver may also update the <*mgmtObj*> resource representation with the retrieved MO information if required according to the local policy.

10) "Create a successful ResponseConfirm".

11) "Send ResponseConfirm primitive".

10.29.3.2 mgmtObjRetrieveResponseConfirm (successful case)

This response is triggered by the MgmtObjRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.350: mgmtObjRetrieveResponseConfirm (successful case)

SCL primitive: mgmtObjRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
mgmtObj	M	Full representation of the retrieved mgmtObj resource

10.29.3.3 mgmtObjRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the MgmtObjRetrieveRequestIndication. The SCL primitive shall comply with table 10.351.

Table 10.351: mgmtObjRetrieveResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.29.4 mgmtObjUpdate

10.29.4.1 mgmtObjUpdateRequestIndication

This request is used to update the full representation of the <*mgmtObj*> resource.

The SCL primitive shall comply with the following.

Table 10.352: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.353: mgmtObjUpdateRequestIndication

SCL Primitive: mgmtObjUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the <*mgmtObj*> resource to be updated
primitiveType	M	MGMTOBJ_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
mgmtObj	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for UPDATE".

If the issuer is a Network Application, the receiver shall perform the following steps, otherwise skip to step 10.

6) "Identify the managed remote entity and the management protocol".

7) "Locate the MO information to be managed on the remote entity".

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive".

10) "Update the addressed resource".

11) "Create a successful response Confirm".

12) "Send response Confirm primitive".

10.29.4.2 mgmtObjUpdateResponseConfirm (successful case)

This response is triggered by the mgmtObjUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.354: mgmtObjUpdateResponseConfirm (successful case)

SCL primitive: mgmtObjUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
mgmtObj	O	Full representation of the updated <*mgmtObj*> resource, if any of the provided attributes were modified by the hosting SCL

10.29.4.3 mgmtObjUpdateResponseConfirm (unsuccessful case)

This response is triggered by the mgmtObjUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.355: mgmtObjUpdateResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.29.5 mgmtObjDelete

10.29.5.1 mgmtObjDeleteRequestIndication

The procedure is used to delete an <*mgmtObj*> resource. The SCL primitive shall comply with the following.

Table 10.356: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.357: mgmtObjDeleteRequestIndication

SCL Primitive: mgmtObjDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the deletion
targetID	M	The URI of the <*mgmtObj*> resource to be deleted
primitiveType	M	MGMTOBJ_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".

If the issuer is a Network Application, the receiver shall perform the following steps, otherwise skip to step 9.

5) "Identify the managed remote entity and the management protocol".
6) "Locate the MO information to be managed on the remote entity".
7) "Establish a management session with the remote entity".
8) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". If the receiver cannot delete the corresponding MO information from the remote entity successfully, it shall still continue to perform step 9 instead of sending an unsuccessful ResponseConfirm primitive.
9) "Delete the addressed resource".
10) "Create a successful ResponseConfirm".
11) "Send ResponseConfirm primitive".

10.29.5.2 mgmtObjDeleteResponseConfirm (successful case)

This response is triggered by the mgmtObjDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.358: mgmtObjDeleteResponseConfirm (successful case)

SCL primitive: mgmtObjDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.29.5.3 mgmtObjDeleteResponseConfirm (unsuccessful case)

This response is triggered by the mgmtObjDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.359: mgmtObjDeleteResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjDeleteResponseConfirm		
Primitive attribute	***Mandatory/Optional***	***Description***
primitiveType	M	MGMTOBJ_DELETE_RESPONSE
errorInfo	M	Provides error information

10.29.6 mgmtObjExecute

10.29.6.1 mgmtObjExecuteRequestIndication

This request is used to trigger the execution of a management command which is represented by the <*mgmtObj*> resource, its sub-resource, or attributes (see clause 10.39).

The SCL primitive shall comply with the following.

Table 10.360: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.361: mgmtObjExecuteRequestIndication

SCL Primitive: mgmtObjExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application originally requesting the updating
targetID	M	The URI of the <*mgmtObj*> resource which represents the command to be executed Or the URI of a <*moAttribute*> attribute(clause 10.39) which represents the command to be executed in the <*mgmtObj*> resource Or the URI provided as the value of the <*moAttribute*> attribute which represents the command to be executed in the <*mgmtObj*> resource
primitiveType	M	MGMTOBJ_EXECUTE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Identify the managed remote entity and the management protocol".

6) "Locate the MO information to be managed on the remote entity".

7) "Establish a management session with the remote entity".

8) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive".

9) "Create a successful response Confirm".

10) "Send response Confirm primitive".

Upon receiving from remote entity a management notification (e.g. OMA-DM "Generic Alert" message [66], BBF-TR-069 "Inform" message (TR-069 Issue 1 Amendment 4 [49]) regarding the execution result or status, the receiver may, as per local policy, update the corresponding <*mgmtObj*> resource or its attribute/sub-resource according to the information provided in the management notification or the information retrieved from the remote entity by sending another management request over mId reference point.

10.29.6.2 mgmtObjExecuteResponseConfirm (successful case)

This response is triggered by the mgmtObjExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.362: mgmtObjExecuteResponseConfirm (successful case)

SCL primitive: mgmtObjExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_EXECUTE_RESPONSE
statusCode	M	STATUS_OK

10.29.6.3 mgmtObjExecuteResponseConfirm (unsuccessful case)

This response is triggered by the mgmtObjExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.363: mgmtObjExecuteResponseConfirm (unsuccessful case)

SCL primitive: mgmtObjExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTOBJ_EXECUTE_RESPONSE
errorInfo	M	Provides error information

10.30 <parameters> resources and management procedures

10.30.1 <parameters> resource

The <parameters> resource shall contain the following attributes referring to sub-resources.

Table 10.364: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
parametersCollection	NP	NP	M	References to sub-resources <*parameters*>, see table 11.37
subscriptionsReference	NP	NP	M	References to sub-resources *subscriptions,* see table 11.37
accessRightID	O	O	O	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
originalMO	O	NP	O	See table 11.36
<moAttribute>	O	O	O	See table 11.36
id	O	NP	M*	The id of the <parameters> resource
NOTE: For the <parameters> resource instances using generic or external data model, there could be zero or multiple <moAttribute> attributes that are optional in createReq, updateReq and response. For ETSI specific <parameters> resource instances, the multiplicity and optionality of the <moAttribute> attribute is described in annex E. A mandatory <moAttribute> shall be provided in the createReq unless a default value has been defined for its absence, otherwise the receiver shall reject the createReq.				

10.30.2 parametersCreate

10.30.2.1 parametersCreateRequestIndication

This primitive creates a new <*parameters*> sub-resource under an <*mgmtObj*> or a <*parameters*> resource. The SCL primitive shall comply with the following.

Table 10.365: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.366: ParametersCreateRequestIndication

SCL Primitive: parametersCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_CREATE_REQUEST
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the <*mgmtObj*> or <*parameters*> resource in which the requested resources shall be created
Resource	**Mandatory/Optional**	**Description**
parameters	M	The parameters resource representation. If the issuer is D/GSCL, the "originalMO" attribute SHALL be provided in the request

The **issuer** shall execute the following steps in order:

1) Primitive specific operation: If the issuer is a D/GSCL of the remote entity, it shall generate the <*parameters*> resource representation based on the MO information of the remote entity to be exposed.

2) "Compose RequestIndication primitive".

3) "Send the RequestIndication to the receiver SCL".

4) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

If the issuer is an M2M Network Application, the receiver shall perform the following steps, otherwise skip to step 11.

6) "Identify the managed remote entity and the management protocol".

7) Primitive specific operation: the receiver shall generate the MO to be added to the remote entity based on the <*parameters*> resource representation provided in the request primitive. The receiver shall determine the target location on the remote entity where the generated MO shall be added based on the relative position of the <*parameters*> resource to its parent <*mgmtObj*> or <*parameters*> resource and the protocol specific data model being used. The receiver may also choose to let the remote entity to decide the target location where the generated MO shall be added using protocol specific mechanism (e.g. 'Inbox' as defined in [50]).

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". If the receiver receives an error response from the remote entity because the MO to be added already exists on the remote entity, the receiver shall check (by e.g. OMA-DM Get command or TR069 GetParameterValues command) if the existing MO is the same as the one to be added, then it shall consider the requested primitive as successfully performed instead of sending an unsuccessful ResponseConfirm primitive; otherwise, it shall reject the request with the "STATUS_CONFLICT" statusCode in the ResponseConfirm primitive. The receiver shall also record the location where the MO is added to the remote entity in the successful case, if the location cannot be determined unambiguously by the name of the <*parameters*> resource afterwards.

10) The receiver may repeat step 9 in order to add to the remote entity the MOs that are mapped from the mandatory sub-resources (including any descendents) that are required to be created automatically with the default attribute values as specified in annex E.

11) "Create the resource". The receiver shall also set the "originalMO" attribute of the created <*parameters*> resource as the location of the corresponding MO on the remote entity as recorded in step 9 or as provided in the <parameters> resource representation in the request primitive.

12) "Create a successful ResponseConfirm". It shall also provide in the response the URI of the new <*parameters*> resource created.

13) "Send Response Confirm primitive".

10.30.2.2 parametersCreateResponseConfirm (successful case)

Confirms the creation of a new <*parameters*> sub-resource under an <*mgmtObj*> or a <*parameters*> resource. The SCL primitive shall comply with the following.

Table 10.367: ParametersCreateResponseConfirm (successful case)

SCL primitive: parametersCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the created resource
Resource	**Mandatory/optional**	**Description**
parameters	O	Full resource representation of the <*parameters*>. This is only present if any of the provided attributes are modified by the hosting SCL

10.30.2.3 ParametersCreateResponseConfirm (unsuccessful case)

This response is triggered by the ParametersCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.368: parametersCreateResponseConfirm (unsuccessful case)

SCL primitive: ParametersCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_CREATE_RESPONSE
errorInfo	M	Provides error information

10.30.3 parametersRetrieve

10.30.3.1 parametersRetrieveRequestIndication

This request is used for retrieving the content of a <*parameters*> resource. It is also possible to retrieve only part of the parameters representation, see clause 10.39.

The SCL primitive shall comply with the following.

Table 10.369: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.370: ParametersRetrieveRequestIndication

SCL Primitive: parametersRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the <*parameters*> resource to be retrieved
primitiveType	M	PARAMETERS_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

If the issuer is an M2M Network Application, the receiver shall perform the following steps, otherwise skip to step 10.

If the addressed resource doesn't contain valid information (e.g. if the resource content is empty or obsolete according to local policy), the receiver shall perform the following steps, otherwise skip to step 10.

6) "Identify the managed remote entity and the management protocol".

7) "Locate the MO information to be managed on the remote entity".

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". The receiver may also update the <*parameters*> resource representation with the retrieved MO information if required according to the local policy.

10) "Create a successful ResponseConfirm".

11) "Send ResponseConfirm primitive".

10.30.3.2 parametersRetrieveResponseConfirm (successful case)

This response is triggered by the ParametersRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.371: ParametersRetrieveResponseConfirm (successful case)

SCL primitive: parametersRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
parameters	M	Full representation of the retrieved parameters resource

10.30.3.3 parametersRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the parametersRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.372: ParametersRetrieveResponseConfirm (unsuccessful case)

SCL primitive: parametersRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.30.4 parametersUpdate

10.30.4.1 parametersUpdateRequestIndication

This request is used to update the full representation of the <parameters> resource. It is also possible to update only part of the <parameters> representation, see clause 10.39.

The SCL primitive shall comply with the following.

Table 10.373: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.374: ParametersUpdateRequestIndication

SCL Primitive: parametersUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the <*parameters*> resource to be updated
primitiveType	M	PARAMETERS_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
parameters	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for UPDATE".

If the issuer is a Network Application, the receiver shall perform the following steps, otherwise skip to step 10.

6) "Identify the managed remote entity and the management protocol".

7) "Locate the MO information to be managed on the remote entity".

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive".

10) "Update the addressed resource".

11) "Create a successful response Confirm".

12) "Send response Confirm primitive".

10.30.4.2 parametersUpdateResponseConfirm (successful case)

This response is triggered by the ParametersUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.375: ParametersUpdateResponseConfirm (successful case)

SCL primitive: ParametersUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
parameters	O	Full representation of the updated parameters resource, if any of the provided attributes are modified by the hosting SCL

10.30.4.3 parametersUpdateResponseConfirm (unsuccessful case)

This response is triggered by the ParametersUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.376: ParametersUpdateResponseConfirm (unsuccessful case)

SCL primitive: ParametersUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.30.5 parametersDelete

10.30.5.1 parametersDeleteRequestIndication

The procedure is used to delete a <*parameters*> resource. The SCL primitive shall comply with the following.

Table 10.377: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.378: ParametersDeleteRequestIndication

SCL Primitive: parametersDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the deletion
targetID	M	The URI of the <*parameters*> resource to be deleted
primitiveType	M	PARAMETERS_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) If the issuer is a Network Application, the receiver shall perform the following steps, otherwise skip to step 9.

6) "Identify the managed remote entity and the management protocol".

7) "Locate the MO information to be managed on the remote entity".

8) "Establish a management session with the remote entity".

9) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive". If the receiver cannot delete the corresponding MO information from the remote entity successfully, it shall still continue to perform step 9 instead of sending an unsuccessful ResponseConfirm primitive.

10) "Delete the addressed resource".

11) "Create a successful ResponseConfirm".

12) "Send ResponseConfirm primitive".

10.30.5.2 parametersDeleteResponseConfirm (successful case)

This response is triggered by the ParametersDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.379: ParametersDeleteResponseConfirm (successful case)

SCL primitive: ParametersDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.30.5.3 parametersDeleteResponseConfirm (unsuccessful case)

This response is triggered by the ParametersDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.380: ParametersDeleteResponseConfirm (unsuccessful case)

SCL primitive: ParametersDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_DELETE_RESPONSE
errorInfo	M	Provides error information

10.30.6 parametersExecute

10.30.6.1 parametersExecuteRequestIndication

This request is used to trigger the execution of a management command which is represented by the <*parameters*> resource, its sub-resource, or attributes (see clause 10.39).

The SCL primitive shall comply with the following.

Table 10.381: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.382: ParametersExecuteRequestIndication

SCL Primitive: parametersExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application originally requesting the updating
TargetID	M	The URI of the <*parameters*> resource which represents the command to be executed Or the URI of a <*moAttribute*> attribute(clause 10.39) which represents the command to be executed in the <*parameters*> resource Or the URI provided as the value of the <*moAttribute*> attribute which represents the command to be executed in the <*parameters*> resource
primitiveType	M	PARAMETERS_EXECUTE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Identify the managed remote entity and the management protocol".

6) "Locate the MO information to be managed on the remote entity".

7) "Establish a management session with the remote entity".

8) "Send the management request(s) to the remote entity corresponding to the received RequestIndication primitive".

9) "Create a successful response Confirm".

10) "Send response Confirm primitive".

Upon receiving from remote entity a management notification (e.g. OMA-DM "Generic Alert" message [66], BBF TR-069 "Inform" message TR-069 Issue 1 Amendment 4 [49]) regarding the execution result or status, the receiver may, as per local policy, update the corresponding <*parameters*> resource or its attribute/sub-resource according to the information provided in the management notification or the information retrieved from the remote entity by sending another management request over mId reference point.

10.30.6.2 parametersExecuteResponseConfirm (successful case)

This response is triggered by the parametersExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.383: ParametersExecuteResponseConfirm (successful case)

SCL primitive: parametersExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_EXECUTE_RESPONSE
statusCode	M	STATUS_OK

10.30.6.3 parametersExecuteResponseConfirm (unsuccessful case)

This response is triggered by the parametersExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.384: ParametersExecuteResponseConfirm (unsuccessful case)

SCL primitive: parametersExecuteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	PARAMETERS_EXECUTE_RESPONSE
errorInfo	M	Provides error information

10.31 <mgmtCmd> resource and management procedures

10.31.1 <mgmtCmd> resource

The <*mgmtCmd*> resource shall contain the following attributes referring to sub-resources.

Table 10.385: Resource description

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
execInstancesReference	NP	NP	M	reference to sub-resource containers, see table 11.37
subscriptionsReference	NP	NP	M	reference to sub-resource subscriptions, see table 11.37
expirationTime	O	O	M*	See table 11.36
accessRightID	O	O	O	See table 11.36
searchStrings	O	O	M	See table 11.36
creationTime	NP	NP	M	See table 11.36
lastModifiedTime	NP	NP	M	See table 11.36
cmdType	M	O	M	See table 11.36
execEnable	M	NP	M	See table 11.36
description	O	O	O	See table 11.36
execReqArgs	O	O	O	See table 11.36
id	O	NP	M*	The id of the <mgmtCmd> resource

10.31.2 mgmtCmdCreate

10.31.2.1 mgmtCmdCreateRequestIndication

This procedure is used for creating an <mgmtCmd> resource. The SCL primitive shall comply with table 10.386. The target SCL will use default values for optional parameters that are not specified.

Table 10.386: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.387: mgmtCmdCreateRequestIndication

SCL Primitive: mgmtCmdCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or D/GSCL which requests the creation of an <*mgmtCmd*> resource
targetID	M	The URI of the target entity where the mgmtCmd resource shall be created This request shall address <sclBase>/scls/<scl>/mgmtObjs <sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/mgmtObjs
primitiveType	M	MGMTCMD_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
mgmtCmd	M	The mgmtCmd resource representation

The issuer shall execute the following steps in order:

1) Primitive specific operation: If the issuer is a D/GSCL (i.e. a remote entity), the issuer shall generate the <mgmtCmd> resource representation based on the information of management commands on the remote entity.
2) "Compose RequestIndication primitive".
3) "Send a RequestIndication to the receiver SCL".
4) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of resource representation for CREATE".
6) "Create the resource".
7) "Create a successful ResponseConfirm".
8) "Send Response Confirm primitive".

10.31.2.2 mgmtCmdCreateResponseConfirm (successful case)

This response is triggered by the mgmtCmdCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.388: mgmtCmdCreateResponseConfirm (successful case)

SCL primitive: mgmtCmdCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_CREATE_RESPONSE
statusCode	M	Provides Success code STATUS_CREATED
resourceURI	M	The URI assigned to the mgmtCmd resource that was created
Resource	**Mandatory/Optional**	**Description**
mgmtCmd	O	Full representation of the created mgmtCmd resource, if any of the provided attributes were modified by the hosting SCL

10.31.2.3 mgmtCmdCreateResponseConfirm (unsuccessful case)

This response is triggered by the mgmtCmdCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.389: mgmtCmdCreateResponseConfirm (unsuccessful case)

SCL primitive: mgmtCmdCreateResponseConfirm		
Primitive Attribute	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_CREATE_RESPONSE
errorInfo	M	Provides Error information

10.31.3 mgmtCmdRetrieve

10.31.3.1 mgmtCmdRetrieveRequestIndication

This request is used for retrieving the content of an mgmtCmd resource. The SCL primitive shall comply with the following.

Table 10.390: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.391: mgmtCmdRetrieveRequestIndication

SCL Primitive: mgmtCmdRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests the retrieving
targetID	M	The URI of the <*mgmtCmd*> resource to be retrieved
primitiveType	M	MGMTCMD_RETRIEVE_REQUEST

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.31.3.2 mgmtCmdRetrieveResponseConfirm (successful case)

This response is triggered by the mgmtCmdRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.392: mgmtCmdRetrieveResponseConfirm (successful case)

SCL primitive: mgmtCmdRetrieveResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
mgmtCmd	M	Full representation of the retrieved mgmtCmd resource

10.31.3.3 mgmtCmdRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the mgmtCmdRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.393: mgmtCmdRetrieveResponseConfirm (unsuccessful case)

SCL primitive: mgmtCmdRetrieveResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.31.4 mgmtCmdUpdate

10.31.4.1 mgmtCmdUpdateRequestIndication

This request is used for updating the full representation of an <*mgmtCmd*> resource or only part of an <*mgmtCmd*> resource using partial addressing (clause 10.39). The SCL primitive shall comply with the following.

Table 10.394: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.395: mgmtCmdUpdateRequestIndication

SCL Primitive: mgmtCmdUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or D/GSCL which requests the retrieving
targetID	M	The URI of the *mgmtCmd*> resource to be updated
primitiveType	M	MGMTCMD_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
mgmtCmd	M	The mgmtCmd resource representation

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Check validity of the resource representation for UPDATE".

6) "Update the addressed resource".

7) "Create a successful response Confirm".

8) "Send response Confirm primitive".

10.31.4.2 mgmtCmdUpdateResponseConfirm (successful case)

This response is triggered by the mgmtCmdUpdateRequestIndication. The SCL primitive shall comply with table 10.396. If the SCL does not accept the attribute values in the mgmtCmdUpdateRequestIndication, the attribute values that are used shall be included in the response.

Table 10.396: mgmtCmdUpdateResponseConfirm (successful case)

SCL primitive: mgmtCmdUpdateResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
mgmtCmd	O	Full representation of the application resource, if any of the provided attributes were modified by the hosting SCL

10.31.4.3 mgmtCmdUpdateResponseConfirm (unsuccessful case)

This response is triggered by the mgmtCmdUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.397: mgmtCmdUpdateResponseConfirm (unsuccessful case)

SCL primitive: mgmtCmdUpdateResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_UPDATE_RESPONSE
errorInfo	M	Provides Error information

10.31.5 mgmtCmdDelete

10.31.5.1 mgmtCmdDeleteRequestIndication

The procedure is used for deleting an mgmtCmd resource. The SCL primitive shall comply with the following.

Table 10.398: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	D/GSCL
Receiver	Local SCL	N/A	NSCL

Table 10.399: mgmtCmdDeleteRequestIndication

SCL Primitive: mgmtCmdDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or D/GSCL which requests the deleting
targetID	M	The URI of the mgmtCmd resource to be deleted
primitiveType	M	MGMTCMD_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".
2) "Send a RequestIndication to the receiver SCL".
3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".

If the issuer is a Network Application and the requested mgmtCmd resource includes pending management commands previously initiated on the remote entity and the pending command is cancellable, the receiver shall perform steps 5-7; otherwise, steps 5-7 shall be skipped.

5) "Identify the managed remote entity and the management protocol".
6) "Establish a management session with the remote entity".
7) "Management Command Conversion and Execution".
8) "Delete the addressed resource".
9) "Create a successful ResponseConfirm".
10) "Send ResponseConfirm primitive".

10.31.5.2 mgmtCmdDeleteResponseConfirm (successful case)

This response is triggered by the mgmtCmdDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.400: mgmtCmdDeleteResponseConfirm (successful case)

SCL primitive: mgmtCmdDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.31.5.3 mgmtCmdDeleteResponseConfirm (unsuccessful case)

This response is triggered by the mgmtCmdDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.401: mgmtCmdDeleteResponseConfirm (unsuccessful case)

SCL primitive: mgmtCmdDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_DELETE_RESPONSE
errorInfo	M	Provides Error information

10.31.6 mgmtCmdExecute

10.31.6.1 mgmtCmdExecuteRequestIndication

The procedure is used for executing a management command represented by an mgmtCmd resource. The SCL primitive shall comply with the following.

Table 10.402: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.403: mgmtCmdExecuteRequestIndication

SCL Primitive: mgmtCmdExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests to execute a management command represented by an <*mgmtCmd*> resource
targetID	M	The URI to address the "*execEnable*" attribute of an <*mgmtCmd*> resource (clause 10.39), or the URI provided as the value of the "*execEnable*" attribute
primitiveType	M	MGMTCMD_EXECUTE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Identify the managed remote entity and the management protocol".

6) "Establish a management session with the remote entity".

7) "Management Command Conversion and Execution".

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.31.6.2 mgmtCmdExecuteResponseConfirm (successful case)

This response is triggered by the mgmtCmdExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.404: mgmtCmdExecuteResponseConfirm (successful case)

SCL primitive: mgmtCmdExecuteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_EXECCUTE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI of the <*execInstance*> resource which is created as a result of command execution

10.31.6.3 mgmtCmdExecuteResponseConfirm (unsuccessful case)

This response is triggered by the mgmtCmdExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.405: mgmtCmdExecuteResponseConfirm (unsuccessful case)

SCL primitive: mgmtCmdExecuteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	MGMTCMD_EXECUTE_RESPONSE
errorInfo	M	Provides Error information

10.32 execInstances resource and management procedures

10.32.1 execInstances resource

The execInstances resource shall contain the following attributes referring to sub-resources.

Table 10.406: Resource description

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
execInstanceCollection	N/A	N/A	M	Reference to sub-resource <execInstance> (see table 9.49 in TS 102 690 [2])
subscriptionsReference	N/A	N/A	M	Reference to sub-resource subscriptions (see table 9.49 in TS 102 690 [2])
creationTime	N/A	N/A	M	See table 9.49 in TS 102 690 [2]
lastModifiedTime	N/A	N/A	M	See table 9.49 in TS 102 690 [2]

The *execInstances* resource is created as a child resource of an <*mgmtCmd*> resource. It is created whenever the <*mgmtCmd*> resource is created. It is deleted when its parent resource <*mgmtCmd*> is deleted. It cannot be updated.

10.32.2 execInstancesCreate

execInstances, as a collection resource, is automatically created as a part of mgmtCmdCreate primitive, which requests to create an <*mgmtCmd*> resource. As a result, this primitive shall not be allowed via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.32.3 execInstancesRetrieve

10.32.3.1 execInstancesRetrieveRequestIndication

This request is used for retrieving the content of an execInstances resource. The SCL primitive shall comply with the following.

Table 10.407: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.408: execInstancesRetrieveRequestIndication

SCL Primitive: execInstancesRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests the retrieving
targetID	M	The URI of the *execInstances* resource to be retrieved
primitiveType	M	EXECINSTANCES_RETRIEVE_REQUEST

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.32.3.2 execInstancesRetrieveResponseConfirm (successful case)

This response is triggered by the execInstancesRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.409: execInstancesRetrieveResponseConfirm (successful case)

SCL primitive: execInstancesRetrieveResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCES_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
execInstances	M	Full representation of the retrieved execInstances resource

10.32.3.3 execInstancesRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the execInstancesRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.410: execInstancesRetrieveResponseConfirm (unsuccessful case)

SCL primitive: execInstancesRetrieveResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCES_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.32.4 execInstancesUpdate

This primitive shall not be allowed via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.32.5 execInstancesDelete

This primitive shall not be allowed via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Cend ResponseConfirm primitive".

10.33 <execInstance> resource and management procedures

10.33.1 <execInstance> resource

The <*execInstance*> resource shall contain the following attributes referring to sub-resources.

Table 10.411: Resource description

AttributeNames	Presence in createReq	Presence in updateReq	Presence in response	Description
subscriptionsReference	N/A	N/A	M	reference to sub-resource subscriptions, see table 11.37
expirationTime	N/A	N/A	M*	See table 11.36
accessRightID	N/A	N/A	O	See table 11.36
creationTime	N/A	N/A	M	See table 11.36
lastModifiedTime	N/A	N/A	M	See table 11.36
execStatus	N/A	N/A	M	See table 11.36
execResult	N/A	N/A	M	See table 11.36
execDisable	N/A	N/A	O	See table 11.36
id	N/A	N/A	M	The id of the <execInstance> resource

10.33.2 execInstanceCreate

An <*execInstance*> will be created automatically by the receiver of mgmtCmdExecute primitive if mgmtCmdExecute is successfully processed by the receiver. As a result, this primitive shall not be allowed via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.33.3 execInstanceRetrieve

10.33.3.1 execInstanceRetrieveRequestIndication

This request is used for retrieving the content of an execInstance resource. The SCL primitive shall comply with the following.

Table 10.412: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.413: execInstanceRetrieveRequestIndication

SCL Primitive: execInstanceRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests the retrieving
targetID	M	The URI of the <*execInstance*> resource to be retrieved
primitiveType	M	EXECINSTANCE_RETRIEVE_REQUEST

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.33.3.2 execInstanceRetrieveResponseConfirm (successful case)

This response is triggered by the execInstanceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.414: execInstanceRetrieveResponseConfirm (successful case)

SCL primitive: execInstanceRetrieveResponseConfirm		
Attribute Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
execInstance	M	Full representation of the retrieved execInstance resource

10.33.3.3 execInstanceRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the execInstanceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.415: execInstanceRetrieveResponseConfirm (unsuccessful case)

SCL primitive: execInstanceRetrieveResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.33.4 execInstanceUpdate

This primitive shall not be allowed via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with the statusCode set to STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.33.5 execInstanceDelete

10.33.5.1 execInstanceDeleteRequestIndication

The procedure is used for deleting an <*execInstance*> resource. The SCL primitive shall comply with the following.

Table 10.416: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.417: execInstanceDeleteRequestIndication

SCL Primitive: execInstanceDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests the deleting
targetID	M	The URI of the <*execInstance*> resource to be deleted
primitiveType	M	EXECINSTANCE_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

If the issuer is an M2M Network Application and the requested execInstance resource represents an unfinished management command previously initiated on the remote entity and this unfinished command is cancellable, the receiver shall perform steps 5-7; otherwise, steps 5-7 shall be skipped.

5) "Identify the managed remote entity and the management protocol".

6) "Establish a management session with the remote entity".

7) "Management Command Conversion and Execution".

8) "Delete the addressed resource".

9) "Create a successful ResponseConfirm".

10) "Send ResponseConfirm primitive".

10.33.5.2 execInstanceDeleteResponseConfirm (successful case)

This response is triggered by the execInstanceDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.418: execInstanceDeleteResponseConfirm (successful case)

SCL primitive: execInstanceDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.33.5.3 execInstanceDeleteResponseConfirm (unsuccessful case)

This response is triggered by the execInstanceDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.419: execInstanceDeleteResponseConfirm (unsuccessful case)

SCL primitive: execInstanceDeleteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_DELETE_RESPONSE
errorInfo	M	Provides Error information

10.33.6 execInstanceExecute

10.33.6.1 execInstanceExecuteRequestIndication

The procedure is used for cancelling a management command represented by an <*execInstance*> resource. The SCL primitive shall comply with the following.

Table 10.420: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	N/A
Receiver	Local SCL	N/A	N/A

Table 10.421: execInstanceExecuteRequestIndication

SCL Primitive: execInstanceExecuteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application which requests to cancel an unfinished management command on the remote entity represented by an <*execInstance*> resource
targetID	M	The URI to address the "*execDisable*" attribute of an <*execInstance*> resource (clause 10.39), or the URI provided as the value of the "*execDisable*" attribute
primitiveType	M	EXECINSTANCE_EXECUTE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of received message".

5) "Identify the managed remote entity and the management protocol".

6) "Establish a management session with the remote entity".

7) "Management Command Conversion and Execution".

8) Primitive specific operation: the receiver shall delete the corresponding execInstance resource.

9) "Create a successful ResponseConfirm".

10) "Send ResponseConfirm primitive".

10.33.6.2 execInstanceExecuteResponseConfirm (successful case)

This response is triggered by the execInstanceExecuteRequestIndication. The SCL primitive shall comply with the following.

Table 10.422: execInstanceExecuteResponseConfirm (successful case)

SCL primitive: execInstanceExecuteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_EXECUTE_RESPONSE
statusCode	M	STATUS_OK

10.33.6.3 execInstanceExecuteResponseConfirm (unsuccessful case)

This response is triggered by the execInstanceExecuteRequestIndication. The SCL primitive shall comply with the following:

Table 10.423: execInstanceExecuteResponseConfirm (unsuccessful case)

SCL primitive: execInstanceExecuteResponseConfirm		
Attributes Name	**Mandatory/Optional**	**Description**
primitiveType	M	EXECINSTANCE_EXECUTE_RESPONSE
errorInfo	M	Provides Error information

10.34 attachedDevices resource and management procedures

10.34.1 attachedDevices resource

The attachedDevices resource shall contain the following sub-resources and attributes.

Table 10.424: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
mgmtObjsReference	N/A	NP	M	URI of mgmtObjs collection resource
attachedDeviceCollection	N/A	NP	M	URI of attachedDevice sub-resources; see table 11.37
subscriptionsReference	N/A	NP	M	URI of subscription collection, see table 11.37
accessRightID	N/A	O	O	See table 11.36
creationTime	N/A	NP	M	See table 11.36
lastModifiedTime	N/A	NP	M	See table 11.36

10.34.2 attachedDevicesCreate

The attachedDevices resource shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.34.3 attachedDevicesRetrieve

10.34.3.1 attachedDevicesRetrieveRequestIndication

This request is used for retrieve the content of an attachedDevices resource. The SCL primitive shall comply with the following.

Table 10.425: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	SCL
Receiver	Local SCL	N/A	Hosting SCL

Table 10.426: attachedDevicesRetrieveRequestIndication

SCL Primitive: attachedDevicesRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the attachedDevices resource to be addressed This URI shall be one of the following: <sclBase>/scls/<scl>/attachedDevices
primitiveType	M	ATTACHEDDEVICES_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of the received message".

5) "Create a collection resource representation".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.34.3.2 attachedDevicesRetrieveResponseConfirm (successful case)

This response is triggered by the attachedDevicesRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.427: attachedDevicesRetrieveResponseConfirm (successful case)

SCL primitive: attachedDevicesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICES_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
attachedDevices	M	List of references(URLs)to all the child resources of the retrieved attachedDevices resource, and all the attributes defined

10.34.3.3 attachedDevicesRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the AttachedDevicesRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.428: attachedDevicesRetrieveResponseConfirm (unsuccessful case)

SCL primitive: attachedDevicesRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICES_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.34.4 attachedDevicesUpdate

10.34.4.1 attachedDevicesUpdateRequestIndication

This request is used for update the content of an attachedDevices resource. The SCL primitive shall comply with the following.

Table 10.429: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	SCL
Receiver	Local SCL	N/A	Hosting SCL

Table 10.430: attachedDevicesUpdateRequestIndication

SCL Primitive: attachedDevicesUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the updating
targetID	M	The URI of the attachedDevices resource to be addressed This URI shall be: <sclBase>/scls/<scl>/attachedDevices
primitiveType	M	ATTACHEDDEVICES_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
attachedDevices	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of the requestingEntity based on accessRightID".
4) "Check the syntax of received message".
5) "Check validity of resource representation for update".
6) "Update the addressed resource".
7) "Create a successful ResponseConfirm".
8) "Send ResponseConfirm primitive".

10.34.4.2 attachedDevicesUpdateResponseConfirm (successful case)

This response is triggered by the AttachedDevicesUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.431: attachedDevicesUpdateResponseConfirm (successful case)

SCL primitive: attachedDevicesUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICES_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
attachedDevices	O	Full representation of the updated attachedDevices resource

10.34.4.3 attachedDevicesUpdateResponseConfirm (unsuccessful case)

This response is triggered by the AttachedDevicesUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.432: attachedDevicesUpdateResponseConfirm (unsuccessful case)

SCL primitive: attachedDevicesUpdateResponseConfirm		
Primitive attribute	***Mandatory/Optional***	***Description***
primitiveType	M	ATTACHEDDEVICES_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.34.5 attachedDevicesDelete

The attachedDevices resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.35 attachedDevice resources and management procedures

10.35.1 attachedDevice resource

The attachedDevice resource shall contain the following sub-resources and attributes.

Table 10.433: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
mgmtObjsReference	NP	NP	M	URI of mgmtObjs collection resource
subscriptionsReference	NP	NP	M	URI of subscriptions collection resource
id	O	NP	M*	Id of the <attachedDevice> in the attachedDevices collection. If the attachedDevice indicated in the request already exists, the hosting SCL may choose a different name
accessRightID	O	O	O	See table 9.53 in TS 102 690 [2]
creationTime	NP	NP	M	See table 9.53 in TS 102 690 [2]
lastModifiedTime	NP	NP	M	See table 9.53 in TS 102 690 [2]

10.35.2 attachedDeviceCreate

10.35.2.1 attachedDeviceCreateRequestIndication

This primitive creates a new <attachedDevice> resource in an attachedDevices collection. The SCL primitive shall comply with the following.

Table 10.434: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	D/GSCL
Receiver	N/A	N/A	NSCL

Table 10.435: attachedDeviceCreateRequestIndication

SCL Primitive: attachedDeviceCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_CREATE_REQUEST
requestingEntity	M	SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created This URI shall be: <sclBase>/scls/<scl>/attachedDevices
Resource	**Mandatory/Optional**	**Description**
attachedDevice	M	The <*attachedDevice*> resource representation

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send the RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) Primitive specific operation: the receiver shall check whether the requestingEntity is the D/GSCL which has registered with the receiver NSCL as the grandparent <scl> resource of the <attachedDevice> resource to be created. If not, the RequestIndication shall be rejected with a STATUS_FORBIDDEN statusCode.

5) "Check the syntax of received message".

6) "Check validity of resource representation for create".

7) "Create the resource".

8) "Create a successful ResponseConfirm".

9) "Send Response Confirm primitive".

10.35.2.2 attachedDeviceCreateResponseConfirm (successful case)

Confirms the creation of a new <*attachedDevice*> resource in an attachedDevices collection. The SCL primitive shall comply with the following.

Table 10.436: attachedDeviceCreateResponseConfirm (successful case)

SCL primitive: attachedDeviceCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
attachedDevice	O	Full resource representation of the <*attachedDevice*>. This is only present if any of the provided attributes where modified by the hosting SCL

10.35.2.3 attachedDeviceCreateResponseConfirm (unsuccessful case)

This response is triggered by the attachedDeviceCreateRequestIndication. The SCL primitive shall comply with the following.

Table 10.437: attachedDeviceCreateResponseConfirm (unsuccessful case)

SCL primitive: attachedDeviceCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_CREATE_RESPONSE
errorInfo	M	Provides error information

10.35.3 attachedDeviceRetrieve

10.35.3.1 attachedDeviceRetrieveRequestIndication

This request is used for retrieve the content of an attachedDevice resource. The SCL primitive shall comply with the following.

Table 10.438: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	N/A	SCL
Receiver	Local SCL	N/A	Hosting SCL

Table 10.439: attachedDeviceRetrieveRequestIndication

SCL Primitive: attachedDeviceRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the <attachedDevice> resource to be addressed
primitiveType	M	ATTACHEDDEVICE_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) "Check the syntax of the received message".

5) "Read the addressed resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.35.3.2 attachedDeviceRetrieveResponseConfirm (successful case)

This response is triggered by the AttachedDeviceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.440: attachedDeviceRetrieveResponseConfirm (successful case)

SCL primitive: attachedDeviceRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
attachedDevice	M	List of references(URLs)to all the child resources of the retrieved attachedDevice resource, and all the attributes defined

10.35.3.3 attachedDeviceRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the AttachedDeviceRetrieveRequestIndication. The SCL primitive shall comply with the following.

Table 10.441: attachedDeviceRetrieveResponseConfirm (unsuccessful case)

SCL primitive: attachedDeviceRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.35.4 attachedDeviceUpdate

10.35.4.1 attachedDeviceUpdateRequestIndication

This request is used for update the content of an attachedDevice resource. The SCL primitive shall comply with the following.

Table 10.442: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	D/GSCL
Receiver	N/A	N/A	NSCL

Table 10.443: attachedDeviceUpdateRequestIndication

SCL Primitive: attachedDeviceUpdateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the updating
targetID	M	The URI of the <attachedDevice> resource to be addressed If an attribute has to be updated, then the URI of the attribute shall be provided
primitiveType	M	ATTACHEDDEVICE_UPDATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
attachedDevice	M	Resource representation

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) Primitive specific operation: the receiver shall check whether the requestingEntity is the D/GSCL which has registered with the receiver NSCL as the grandparent <scl> resource of the <attachedDevice> resource to be updated. If not, the RequestIndication shall be rejected with a STATUS_FORBIDDEN statusCode.

5) "Check the syntax of received message".

6) "Check validity of resource representation for update".

7) "Update the addressed resource".

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.35.4.2 attachedDeviceUpdateResponseConfirm (successful case)

This response is triggered by the AttachedDeviceUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.444: attachedDeviceUpdateResponseConfirm (successful case)

SCL primitive: attachedDeviceUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_UPDATE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
attachedDevice	O	Full representation of the updated attachedDevice resource

10.35.4.3 attachedDeviceUpdateResponseConfirm (unsuccessful case)

This response is triggered by the AttachedDeviceUpdateRequestIndication. The SCL primitive shall comply with the following.

Table 10.445: attachedDeviceUpdateResponseConfirm (unsuccessful case)

SCL primitive: attachedDeviceUpdateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_UPDATE_RESPONSE
errorInfo	M	Provides error information

10.35.5 attachedDeviceDelete

10.35.5.1 attachedDeviceDeleteRequestIndication

The procedure is used to delete a <*attachedDevice*> resource. The SCL primitive shall comply with the following.

Table 10.446: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	N/A	N/A	D/GSCL
Receiver	N/A	N/A	NSCL

Table 10.447: attachedDeviceDeleteRequestIndication

SCL Primitive: attachedDeviceDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	SCL originally requesting the deletion
targetID	M	The URI of the <*attachedDevice*> resource to be deleted
primitiveType	M	ATTACHEDDEVICE_DELETE_REQUEST

The issuer shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of the requestingEntity based on accessRightID".

4) Primitive specific operation: the receiver shall check whether the requestingEntity is the D/GSCL which has registered with the receiver NSCL as the grandparent <scl> resource of the <attachedDevice> resource to be deleted. If not, the RequestIndication shall be rejected with a STATUS_FORBIDDEN statusCode.

5) "Check the syntax of received message".

6) "Delete the addressed resource".

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.35.5.2 attachedDeviceDeleteResponseConfirm (successful case)

This response is triggered by the AttachedDeviceDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.448: attachedDeviceDeleteResponseConfirm (successful case)

SCL primitive: attachedDeviceDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.35.5.3 attachedDeviceDeleteResponseConfirm (unsuccessful case)

This response is triggered by the AttachedDeviceDeleteRequestIndication. The SCL primitive shall comply with the following.

Table 10.449: attachedDeviceDeleteResponseConfirm (unsuccessful case)

SCL primitive: attachedDeviceDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	ATTACHEDDEVICE_DELETE_RESPONSE
errorInfo	M	Provides error information

10.36 notificationChannels resource and management procedures

10.36.1 notificationChannels resource

The notificationChannels resource shall contain the following sub-resources and attributes.

Table 10.450: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
notificationChannelCollection	NP	NP	M	
creationTime	NP	NP	M	
lastModifiedTime	NP	NP	M	

10.36.2 notificationChannelsCreate

The *notificationChannels* collection resource shall not be created directly via the API. It is created whenever the parent <*scl*> resource is created or when its parent <*application*> resource is created. The *accessRightID* shall be initialized to same value as the *accessRightID* in the parent.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.36.3 notificationChannelsRetrieve

10.36.3.1 notificationChannelsRetrieveRequestIndication

This primitive is used to retrieve an scls collection Resource. The SCL primitive shall comply with the following.

Table 10.451: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.452: notificationChannelsRetrieveRequestIndication

SCL Primitive: notificationChannelsRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the primitive
targetID	M	Indicates the notificationChannels collection resource to be retrieved
primitiveType	M	NOTIFICATION_CHANNELS_RETRIEVE_REQUEST

The issuer shall execute the following steps in order:

1) "Compose Request indication primitive".

2) "Send RequestIndication primitive".

3) "Wait for ResponseConfirm".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting" The receiver shall not execute following steps if the request is re-targeted.
2) "Check existence of the addressed resource".
3) "Check authorization of requestingEntity based on default access rights".
4) "Check the syntax of the received message".
5) "Create a collection resource representation".
6) "Create a successful ResponseConfirm" with the created collection resource representation.
7) "Send ResponseConfirm".

10.36.3.2 notificationChannelsRetrieveResponseConfirm (successful case)

Confirms the retrieval of an notificationChannels Resource. The SCL primitive shall comply with the following.

Table 10.453: notificationChannelsRetrieveResponseConfirm (successful case)

SCL primitive: notificationChannelsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNELS_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
notificationChannels	M	The notificationChannels resource representation as indicated in clause 10.36.1

10.36.3.3 notificationChannelsRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the notificationChannelsRetrieveRequestIndication. The SCL primitive shall comply with table 10.454.

Table 10.454: notificationChannelsRetrieveResponseConfirm (unsuccessful case)

SCL primitive: notificationChannelsRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNELS_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.36.4 notificationChannelsUpdate

The notificationChannels collection resource shall not be updated, since it does not have any attributes that are modifiable in an update.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.36.5 notificationChannelsDelete

The notificationChannels collection resource shall not be deleted directly via the API. It is deleted whenever the parent sclBase resource is deleted or when its parent scl resource is deleted.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.37 <notificationChannel> resource and management procedures

10.37.1 <notificationChannel> resource

The notificationChannel resource shall contain the following attributes referring to sub-resources.

Table 10.455: Resource description

AttributeName	Presence in createReq	Presence in updateReq	Presence in response	Description
id	O	NP	M*	When used in a CREATE request, this is interpreted as a suggested identification of the resource in the collection resource. When returned in a response, this is the actual identification chosen by the hosting SCL for use in its collection. This may be different from the requested identity. See table 11.36.
channelType	M	NP	M	Specifies the type of notification channel to be used (i.e. method that will be used to report new notifications on the channel).
contactURI	NP	NP	M	Specified by the hosting SCL. Contains a URI used when establishing subscriptions for notifications. See table 11.36.
channelData	NP	NP	M	Contains specific information for the notification channel type specified in channelType. In the present document only long-polling type is supported and the channel data is defined in the type LongPollingData (see clause 11.4) which is derived from ChannelType. See table 11.36.
creationTime	NP	NP	M	See table 11.36.
lastModifiedTime	NP	NP	M	See table 11.36.

10.37.2 notificationChannelCreate

10.37.2.1 notificationChannelCreateRequestIndication

This request is used to create a new <notificationChannel> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.456: Applicability of the primitive

Primitive applicability			
Applicable interfaces	mIa	dIa	mId
Issuer	Application	Application on D'	SCL
Receiver	Local SCL	Local SCL	Hosting SCL

Table 10.457: NotificationChannelCreateRequestIndication

SCL Primitive: notificationChannelCreateRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the creation
targetID	M	Indicates the collection resource in which the requested resources shall be created
primitiveType	M	NOTIFICATION_CHANNEL_CREATE_REQUEST
Resource	**Mandatory/Optional**	**Description**
notificationChannel	O	The representation of the resource to be created

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource". In this case the addressed resource shall be the provided collection resource.

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Check validity of resource representation for create".

6) "Create the resource".

7) Primitive specific operation: if the channelType indicated in the CREATE request is LONG_POLLING, then the hosting SCL shall:

   a) Create two URIs; one contactURI, which is returned as the contactURI in the resource representation. And one longPollingURI, which is returned in the channelData attribute. The handling of these URIs is described separately in clauses 10.37.6 and 10.37.7.

   b) Set the internal state of the notificationChannel to PAUSED. Note that this state is not exposed via the API, but mainly used for internal bookkeeping in the hosting SCL. See clause 10.37.7.7 for the state transition diagram of the notification channel resource.

   c) If the hosting SCL is the NSCL, then he <scl> ancestor of the notificationChannel resource is updated to maintain the following invariant: The onlineStatus of the <scl> resource is ONLINE if there is at least one m2mPoc resource in its collection with an onlineStatus set to ONLINE or if there is long polling active for the <scl> (the internal state some notificationChannel resource residing under the <scl> resource is ACTIVE (i.e. POLLING or PAUSED). The onlineStatus of the <scl> resource is NOT_REACHABLE if all the m2mPoc resources in its collection have an onlineStatus set to NOT_REACHABLE. The onlineStatus of the <scl> resource is OFFLINE in all other cases (i.e. if there are no m2mPoc resources in the collection or if all m2mPoc resource in the collection have their onlineStatus attribute set to OFFLINE).

   d) start timer T1 for the resource. The value is initialized according to server policy. This is also a timer used for internal bookkeeping and it is not exposed via the API.

   e) Start timer T2 for the resource. The value is initialized according to server policy. This is also a timer used for internal bookkeeping and it is not exposed via the API.

8) "Create a successful ResponseConfirm".

9) "Send ResponseConfirm primitive".

10.37.2.2 notificationChannelCreateResponseConfirm (successful case)

It confirms the creation of a new <notificationChannel> resource in a Service Capability Layer. The SCL primitive shall comply with the following.

Table 10.458: NotificationChannelCreateResponseConfirm (successful case)

SCL primitive: NotificationChannelCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_CREATE_RESPONSE
statusCode	M	STATUS_CREATED
resourceURI	M	URI assigned to the resource
Resource	**Mandatory/Optional**	**Description**
notificationChannel	M	then the complete content of the resource as described above is returned in the response

10.37.2.3 notificationChannelCreateResponseConfirm (unsuccessful case)

This response is triggered by the notificationChannelCreateRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.459: NotificationChannelCreateResponseConfirm (unsuccessful case)

SCL primitive: notificationChannelCreateResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_CREATE_RESPONSE
errorInfo	M	Provides error information

10.37.3 notificationChannelRetrieve

10.37.3.1 notificationChannelRetrieveRequestIndication

This request is used for retrieving the content of an <notificationChannel> resource. The SCL primitive shall comply with the following.

Table 10.460: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application on D'	SCL
Receiver	local SCL	Local SCL	local SCL

Table 10.461: notificationChannelRetrieveRequestIndication

SCL Primitive: notificationChannelRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the retrieving of the resource
targetID	M	The URI of the resource to be retrieved
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Read the resource".

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

10.37.3.2 notificationChannelRetrieveResponseConfirm (successful case)

This response is triggered by the notificationChannelRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.462: NotificationChannelRetrieveResponseConfirm (successful case)

SCL primitive: notificationChannelRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
notificationChannel	M	Complete content of the resource is returned in the response

10.37.3.3 notificationChannelRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the NotificationChannelRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.463: NotificationChannelRetrieveResponseConfirm (unsuccessful case)

SCL primitive: notificationChannelRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.37.4 notificationChannelUpdate

The notification channel shall not be updated via the API.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.37.5 notificationChannelDelete

10.37.5.1 notificationChannelDeleteRequestIndication

The procedure is used to delete an <notificationChannel> resource. The SCL primitive shall comply with the following.

Table 10.464: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application on D'	SCL
Receiver	local SCL	local SCL	Hosting SCL

Table 10.465: NotificationChannelDeleteRequestIndication

SCL Primitive: notificationChannelDeleteRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the resource deletion
targetID	M	The URI of the resource to be deleted
primitiveType	M	NOTIFICATION_CHANNEL_DELETE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource".

3) "Check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) "Delete the resource".

6) Primitive specific operation: when the resource is deleted also the related longPollingURI and contactURI will become invalid. Any attempt to address those URIs will be handled as if the resource does not exist. Any pending notifications will be rejected with STATUS_NOT_FOUND. All associated timers are stopped. If the hosting SCL is the NSCL, then the <scl> ancestor of the notificationChannel resource is updated to maintain the following invariant: The onlineStatus of the <scl> resource is ONLINE if there is at least one m2mPoc resource in its collection with an onlineStatus set to ONLINE or if there is long polling active for the <scl> (the internal state some notificationChannel resource residing under the <scl> resource is ACTIVE (i.e. POLLING or PAUSED). The onlineStatus of the <scl> resource is NOT_REACHABLE if all the m2mPoc resources in its collection have an onlineStatus set to NOT_REACHABLE. The onlineStatus of the <scl> resource is OFFLINE in all other cases (i.e. if there are no m2mPoc resources in the collection or if all m2mPoc resource in the collection have their onlineStatus attribute set to OFFLINE).

7) "Create a successful ResponseConfirm".

8) "Send ResponseConfirm primitive".

10.37.5.2 notificationChannelDeleteResponseConfirm (successful case)

This response is triggered by the notificationChannelDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.466: NotificationChannelDeleteResponseConfirm (successful case)

SCL primitive: notificationChannelDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_DELETE_RESPONSE
statusCode	M	STATUS_OK

10.37.5.3 notificationChannelDeleteResponseConfirm (unsuccessful case)

This response is triggered by the notificationChannelDeleteRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.467: NotificationChannelDeleteResponseConfirm (unsuccessful case)

SCL primitive: NotificationChannelDeleteResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_DELETE_RESPONSE
errorInfo	M	Provides error information

10.37.6 Long polling

Long polling is the access of the notificationChannel on its exposed longPollingURI that is returned to the creator as part of the channelData for notificationChannels with a channelType of LONG_POLLING.

This clause describes what happens when the long polling URI is RETRIEVED. Below this is treated as a notification channel RETRIEVE request with a targetID set to the longPollingURI exposed by the notificationChannel resource.

Some best practices and known issues with relation to long polling can be found in RFC 6202 [i.3].

10.37.6.1 notificationChannelCreateRequestIndication

A CREATE where the targetID the URI returned as the longPollingURI element in the attribute channelData of the corresponding notificationChannel resource, shall be rejected. The longPollingURI is created together with notificationChannel with channelType LONG_POLLING.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.37.6.2 notificationChannelRetrieveRequestIndication

This request is used for performing a long poll. The SCL primitive shall comply with the following.

Table 10.468: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application on D'	SCL
Receiver	Local SCL	Local SCL	local SCL

Table 10.469: NotificationChannelRetrieveRequestIndication

SCL Primitive: notificationChannelRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL originally requesting the long polling
targetID	M	The URI returned as the longPollingURI element in the attribute channelData of the corresponding notificationChannel resource
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_REQUEST

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication primitive".

2) "Send a RequestIndication to the receiver SCL".

3) "Wait for ResponseConfirm primitive".

4) If the ResponseConfirm does not contain a notification, i.e. does not contain any resource representation, the issuer starts again at step 1.

5) If the ResponseConfirm includes a notification, then the notification is handled locally just like an asynchronously received notification. The issuer starts again at step 1.

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource". I.e. in this case the existence of the notificationChannel resource that corresponds to the longPollingURI is checked.

3) "check authorization of requestingEntity based on default access rights".

4) "Check the syntax of received message".

5) Primitive specific operations:

   a) Change the internal state of the notificationChannel to POLLING.

   b) If the hosting SCL is the NSCL then the <scl> ancestor of the notificationChannel resource is updated to maintain the following invariant: The onlineStatus of the <scl> resource is ONLINE if there is at least one m2mPoc resource in its collection with an onlineStatus set to ONLINE or if there is long polling active for the <scl> (the internal state some notificationChannel resource residing under the <scl> resource is ACTIVE (i.e. POLLING or PAUSED). The onlineStatus of the <scl> resource is NOT_REACHABLE if all the m2mPoc resources in its collection have an onlineStatus set to NOT_REACHABLE. The onlineStatus of the <scl> resource is OFFLINE in all other cases (i.e. if there are no m2mPoc resources in the collection or if all m2mPoc resource in the collection have their onlineStatus attribute set to OFFLINE).

   c) Stop timer T2 if it was active.

   d) Start timer T3.

   e) If there is already one or more pending notifications for the associated notification channel resource, then:

      i) The state of the notification channel is changed to PAUSED.

      ii) Timer T3 is stopped.

      iii) Timer T2 is started.

      iv) The first of these pending notifications is used for the ResponseConfirm in step 6.

      v) A successful ResponseConfirm is sent for the pending notification as described in clause 10.2.2.

f) If there is no pending notification for the associated notification channel resource, then the hosting SCL shall wait for either T3 to expire, T1 to expire or for a notification received on the associated contactURI:

i) If T3 expires, the state is changed to PAUSED. Continue with step 6, where there will be no content included in the ResponseConfirm.

ii) If T1 expires, the resource is deleted according to the actions "delete addressed resource". The request is rejected with a STATUS_NOT_FOUND.

iii) The case a notification is received the handling shall be as in clause 10.37.7.

6) "Create a successful ResponseConfirm".

7) "Send ResponseConfirm primitive".

Post primitive actions:

If timer T2 expires, then the state of the notificationChannel shall:

8) Change its internal state to INACTIVE.

9) If the hosting SCL is the NSCL then the <scl> ancestor of the notificationChannel resource is updated to maintain the following invariant: The onlineStatus of the <scl> resource is ONLINE if there is at least one m2mPoc resource in its collection with an onlineStatus set to ONLINE or if there is long polling active for the <scl> (the internal state some notificationChannel resource residing under the <scl> resource is ACTIVE (i.e. POLLING or PAUSED). The onlineStatus of the <scl> resource is NOT_REACHABLE if all the m2mPoc resources in its collection have an onlineStatus set to NOT_REACHABLE. The onlineStatus of the <scl> resource is OFFLINE in all other cases (i.e. if there are no m2mPoc resources in the collection or if all m2mPoc resource in the collection have their onlineStatus attribute set to OFFLINE).

10.37.6.3 notificationChannelRetrieveResponseConfirm (successful case)

This response is triggered by the notificationChannelRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.470: notificationChannelRetrieveResponseConfirm (successful case)

SCL primitive: notificationChannelRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
notify	O	The received notification, if any. If no notification is received, this means that timer T1 has expired and that the issuer shall re-issue the long polling request

10.37.6.4 notificationChannelRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the notificationChannelRetrieveRequestIndication primitive. The SCL primitive shall comply with table 10.471.

Table 10.471: NotificationChannelRetrieveResponseConfirm (unsuccessful case)

SCL primitive: notificationChannelRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.37.6.5 notificationChannelUpdateRequestIndication

A UPDATE where the targetID the URI returned as the longPollingURI element in the attribute channelData of the corresponding notificationChannel resource, shall be rejected.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.37.6.6 notificationChannelDeleteRequestIndication

A DELETE where the targetID the URI returned as the longPollingURI element in the attribute channelData of the corresponding notificationChannel resource, shall be rejected. The longPollingURI shall only be deleted when the corresponding notificationChannel is deleted.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.37.7 Receive notification

This clause describes what happens when a notify is sent to contactURI of a notificationChannel. Below this is treated as a notification channel NOTIFY request with a targetID set to the contactURI exposed by the notificationChannel resource.

10.37.7.1 notificationChannelCreateRequestIndication

A UPDATE where the targetID the URI returned as the contactURI element returned in the notificationChannel resource, shall be rejected.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.
2) "Send ResponseConfirm primitive".

10.37.7.2 notificationChannelNotifyRequestIndication

This request is used for receiving a notify request of an <notificationChannel> resource. The SCL primitive shall comply with the following.

Table 10.472: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	SCL	SCL	local SCL

Table 10.473: NotificationChannelNotifyRequestIndication

SCL Primitive: notificationChannelNotifyRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL sending the notify
targetID	M	The URI returned as the contactURI attribute of the corresponding notificationChannel resource
primitiveType	M	NOTIFICATION_CHANNEL_NOTIFY_REQUEST

From the **issuer** point of view this shall treated like a subscriptionNotifyRequestIndication.

The **receiver** shall execute the following steps in order. In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm" and then "Send ResponseConfirm primitive". The corresponding statusCode as indicated in clause 11.3 shall be included in the ResponseConfirm primitive.

1) "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check existence of the addressed resource". I.e. in this case the existence of the notificationChannel resource that corresponds to the contactURI is checked.

3) "Check the syntax of received message".

4) Primitive specific operations:

    a) If the state of the corresponding notificationChannel resource is PAUSED or INACTIVE, then the notification shall become pending for this notificationChannel:

        i) The hosting SCL may limit the amount of pending notifications for a channel according to server policy. If the limit is reached the request is rejected with a STATUS_NOT_AVAILBLE.

        ii) The hosting SCL starts a timer to wait for the corresponding long polling request, the pending notify timer.

        iii) If a long polling request is received, this is handled as described in clause 10.37.6. The timer is stopped.

        iv) If the pending notify timer expires, the notify request is rejected with a STATUS_TIMEOUT.

    b) If the state of the corresponding notificationChannel resource is POLLING, then the handling is as described in clause "long polling request".

5) "Create a successful ResponseConfirm".

6) "Send ResponseConfirm primitive".

10.37.7.3 notificationChannelNotifyResponseConfirm (successful case)

This response is triggered by the notificationChannelNotifyRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.474: NotificationChannelNotifyResponseConfirm (successful case)

SCL primitive: notificationChannelNotifyResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_NOTIFY_RESPONSE
statusCode	M	STATUS_OK

10.37.7.4 notificationChannelRetrieveResponseConfirm (unsuccessful case)

This response is triggered by the NotificationChannelRetrieveRequestIndication primitive. The SCL primitive shall comply with the following.

Table 10.475: NotificationChannelNotifyResponseConfirm (unsuccessful case)

SCL primitive: notificationChannelRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE
errorInfo	M	Provides error information

10.37.7.5 notificationChannelUpdateRequestIndication

A UPDATE where the targetID the URI returned as the contactURI element returned in the notificationChannel resource, shall be rejected.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.37.7.6 notificationChannelDeleteRequestIndication

A DELETE where the targetID the URI returned as the longPollingURI element in the attribute channelData of the corresponding notificationChannel resource, shall be rejected. The longPollingURI shall only be deleted when the corresponding notificationChannel is deleted.

The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.37.7.7 notificationChannel statemachine

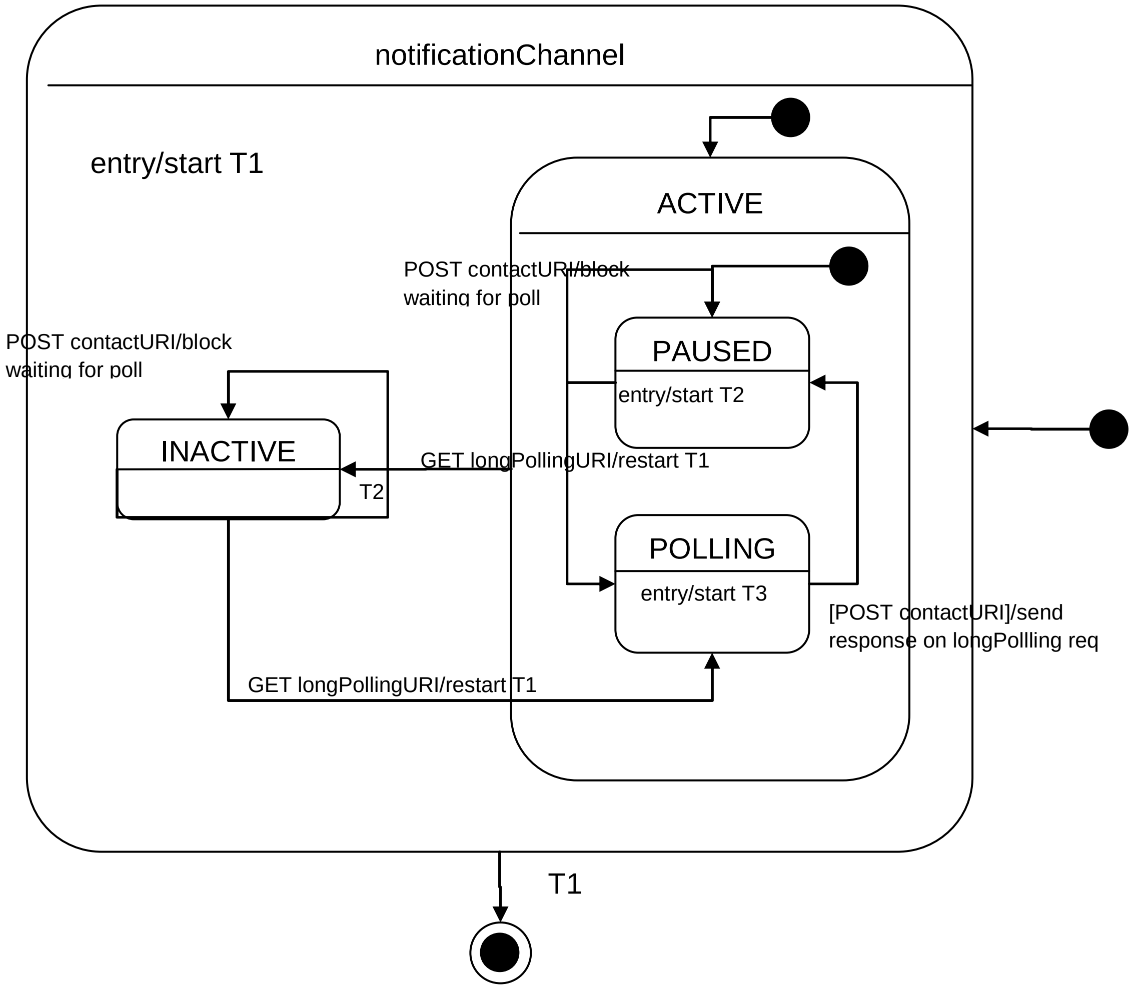


Figure 10.2

The following states are defined:

- INACTIVE - no long polling is active at the moment.
- ACTIVE - long polling is active at the moment. This means there is either a long polling request outstanding or there has been a long polling request is expected soon. ACTIVE consists of two sub states PAUSED and POLLING.
- POLLING - the hosting has received a long polling request, but not responded to it yet. Incoming notifications be forwarded to the long polling issuer as a response in this state.
- PAUSED - the channel was just created or the hosting SCL responded very recently on a long polling request.

The following timers are identified. All these timers are implementation specific:

- T1 - the lifetime of the notification channel. This timer is restarted whenever a new long polling request is received.
- T2 - the time the hosting SCL will still consider long polling active after a response was sent on a previous long polling request, or when the notification channel was created and before the first long polling request is received.
- T3 - the time the hosting SCL will wait before responding with an empty response to the long polling request in case no incoming notifications are received during a long polling request.

10.38 discovery resource and management procedures

10.38.1 discovery resource

The discovery resource shall be used to perform a discovery on the sclBase and provide a result back to the issuer.

The discovery resource does not represent a real resource in the sense that its representation does not have an e-tag. Since discovery resource is a virtual resource, resource Create, Update and Delete requests shall not be allowed to perform.

10.38.2 discoveryCreate

The discovery resource shall not be created via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.38.3 discoveryRetrieve

10.38.3.1 discoveryRetrieveRequestIndication

discoveryRetrieveRequestIndication is a request to perform Resource Discovery on the Hosting SCL. The request is addressed to <sclBase>/discovery of the Hosting SCL. A hosting SCL indicated in the table may be a local SCL as well. The primitive shall comply with the following.

Table 10.476: Applicability of the primitive

Primitive applicability			
Applicable interfaces	**mIa**	**dIa**	**mId**
Issuer	Application	Application	SCL
Receiver	Hosting SCL	Hosting SCL	Hosting SCL

Table 10.477: DiscoveryRetrieveRequestIndication

SCL Primitive :discoveryRetrieveRequestIndication		
Primitive attribute	**Mandatory/Optional**	**Description**
requestingEntity	M	Application or SCL requesting the search
targetID	M	Hosting SCL where the resource discovery to be executed. Discovery is done through a well-known directory <sclBase>/discovery
primitiveType	M	Indicate the type of primitive: DISCOVERY_RETRIEVE _REQUEST
searchPrefix	O	The path under which the discovery procedure shall execute. If this field is not specified, discovery will be done under <sclBase>/
maxSize	O	This value specifies the maximum number of discovered resources that may be returned If this value is not specified, then the receiver will assume that the sender desires no maximum? The receiver may override this value with a smaller maxSize
filterCriteria	O	The filterCriteria specifies what conditions a resource shall meet to be discovered. The encoding of the filterCriteria is specified in clause 11.4 If no filterCriteria is provided, everything under the discovery resource shall be retrieved. maxSize may be used to limit the size

Procedure description

The **issuer** shall execute the following steps in order:

1) "Compose RequestIndication Primitive".

2) "Send a RequestIndication to the receiver SCL entity".

3) "Wait for ResponseConfirm primitive".

The **receiver** shall execute the following steps in order.

In case of error in any of the steps below, the receiver shall execute "Create an unsuccessful ResponseConfirm".

1) The corresponding statusCode as indicated in enumeration StatusCode in clause 11.3 shall be included in the ResponseConfirm. "Re-targeting". The receiver shall not execute following steps if the request is re-targeted.

2) "Check the existence of the addressed resource".

3) "Check the syntax of the received message".

4) Primitive specific operation: "Retrieve the URIs of the resources that matched to the filter criteria".

   - As default behaviour the discovery procedure shall identify all matching resources from the entire hierarchy under sclBase. Optionally the Issuer may specify a prefix under which the discovery is performed.

   - If the Issuer specifies filter criteria, the discovery procedure shall identify matching resources under sclBase matching to the filter criteria.

   - For any application resources under the search prefix, the discovery procedure shall identify any elements in their aPoCPaths attribute which has a searchString in their searchStrings element that matches the specified filterCriteria. If an aPoCPath element does not have a searchStrings element, then the aPocPath element shall be identified only if that application resource matches the filterCriteria. The URI aggregated by this procedure shall be the URI of the corresponding application resource with the path-element contained the matching aPocPath appended to it.

   - Then the Hosting SCL shall build the corresponding discoveyURI of the URI of all the matched resources that the issuer has Discovery access right.

   - If announced resources match the filter criteria, the returned discoveryURI will be the value of link attribute of the announced resources, which points to the location of the original resource.

- The hosting SCL shall determine what is the maximum number of resource URIs it shall send in a response. The hosting SCL may use the maxSize attribute in the request as an indication of the capabilities of the issuer. The hosting SCL shall never send more resource URIs in the response then is indicated by the issuer, but it may send less.

- If the number of matching resources exceeds the number of resource URIs the hosting SCL is willing to send the hosting SCL shall set the truncated attribute to TRUE and it shall set the matchSize to the total number of matching resources.

5) "Create a successful ResponseConfirm".

The response shall contain the matched resources' URI list.

10.38.3.2 discoveryRetrieveResponseConfirm (successful case)

DiscoveryRetrieveResponseConfirm is the response to a request to perform resource discovery in an SCL. A successful case is a case where the discovery procedure was successfully completed, regardless of whether or not a resource matching the filer criteria was discovered. The SCL primitive shall comply with the following.

Table 10.478: DiscoveryRetrieveResponseConfirm (successful case)

SCL primitive: discoveryRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	DISCOVERY_RETRIEVE_RESPONSE
statusCode	M	STATUS_OK
Resource	**Mandatory/Optional**	**Description**
discovery	M	This is a representation of the discovery result

The discovery resource representation shall be in the following structure and be sent to the issuer.

Table 10.479: Discovery response entity

Attribute	Mandatory/Optional	Description
matchSize	O	The number of resources that matched the filterCriteria
truncated	O	This field is used to indicate that there are URI(s) that match the filterCriteria but are not included in the discovery. response If not included, this shall be interpreted as if the value was FALSE
discoveryURI	M	The list of URI(s) of the resources that matched the discovery request. It could be an empty list

10.38.3.3 discoveryRetrieveResponseConfirm (unsuccessful case)

This is in response to a request to perform resource discovery in an SCL. An unsuccessful case is a case where the discovery procedure was not successfully completed. The SCL primitive shall comply with the following.

Table 10.480: DiscoveryRetrieveResponseConfirm (unsuccessful case)

SCL primitive: discoveryRetrieveResponseConfirm		
Primitive attribute	**Mandatory/Optional**	**Description**
primitiveType	M	DISCOVERY_RETRIEVE_RESPONSE
errorInfo	M	Provides Error information

10.38.4 discoveryUpdate

The discovery resource shall not be updated via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.38.5 discoveryDelete

The discovery resource shall not be deleted via the API. The **receiver** shall:

1) "Create an unsuccessful ResponseConfirm" with statusCode STATUS_METHOD_NOT_ALLOWED.

2) "Send ResponseConfirm primitive".

10.39 Partial Addressing

Partial addressing allows the addressing of a specific attribute or part of attribute in a resource and perform REST methods on it.

This allows for the following:

- Partial retrieve:
  - Retrieval of a specific attribute or element (RETRIEVE).
  - Check if an attribute or element has a specific value. This is only possible if the addressed attribute or element is of a basic type or enumerations (RETRIEVE).
- Partial update:
  - Deleting of a specific (O - optional) attribute or deleting a element (DELETE).
  - Reverting of a mandatory (M* - mandatory) attribute to default value (DELETE).
  - Replacing of a specific attribute or element with a new value (UPDATE).
  - Adding a (O - optional) attribute to the resource (CREATE/UPDATE).
  - Adding an optional (0..1) element (CREATE/UPDATE).
  - Adding a value to a collection attribute or a collection element (CREATE).
  - Adding multiple values to a collection attribute or collection element (CREATE).
- Partial subscribe (and corresponding partial notify) (CREATE and NOTIFY respectively):
  - Subscribe to a specific attribute or element modification.

As is obvious from the list, there is no partial create and no partial delete, since creating or deleting a part of a resource actually is a partial modify when viewing it on the resource level.

10.39.1 Attributes/Element Types

We shall distinguish different types of attributes or elements.

Basic type: the attribute or element is of a type listed in clause 11.2.

EXAMPLE 1: Attribute lastModifiedTime has type DateTime.
Element member in the MemberList type has type AnyURI.

Enum: the attribute of element has a type that is listed in the clause 11.3.

EXAMPLE 2: Attribute onlineStatus has a type OnlineStatus that is an enumeration.

Structure Type: the attribute or element has a complex type listed in clause 11.4 with the following restrictions; each element in the complexType has a multiplicity of 1 or 0..1.

EXAMPLE 3: Attribute announceTo is a structured type.

Collection Type: the attribute or element has a complex type listed in clause 11.4 with the following restrictions; there is only one element in the complexType and it has a multiplicity of 0..unbounded or 1..unbounded.

EXAMPLE 4: Attribute permissions is a collection type.

Collection of complex Type: a collection type, where the element has a complex type.

EXAMPLE 5: Attribute permissions is a collection of complex type.

Collection of basic Type: acollection type, where the element has a basic type.

EXAMPLE 6: Attribute members is a collection of basic type.

Collection of Enum: a collection type, where the element has a enum type.

EXAMPLE 7: Element permissionFlags is a collection of enum type.

10.39.2 Partial Accessor

The targetID in the request primitive normally is the URI that identifies the resource.

Partial addressing extends this addressing scheme with a "partialAccessor", so instead of only identifying the resource, it shall allow identification of a specific attribute or part of an attribute (element) belonging to that resource.

For example, the resource URI of an accessRight resource could be http://m2m.operator.org/accessRights/myAR.

The attribute accessor for the a specific permission in this AR would be "/permissions/myPermission".

To address this specific permission in the accessRight resource the following URI can be used http://m2m.operator.org/accessRights/myAR/permissions/myPermission".

The partialAccessor identifies a specific attribute or element or a specific value of an attribute or element.

The partialAccessor shall be defined recursively.

The partialAccessor "/" shall be the same as addressing the resource representation itself (i.e. this is the same as just specifying the resource URI without a partialAccessor). The resource representation is a structured type where each attribute name of an element.

attributeAccessor

If the partialAccessor <prefix> identifies a resource representation, the partialAccessor "<prefix>/<name>" shall identify the resource attribute. The addressed attribute (<name>) shall correspond to the 'attribute name" column in the resource table corresponding to the primitive.

elementAccessor

If the partialAccessor <prefix> identifies a structured type, the partialAccessor "<prefix>/<name>" shall identify the element or element in structured type. The addressed element corresponds to the 'name' column in the structured type as shown in the table in clause 11.4.

memberAccessor, basic

If the partialAccessor <prefix> identifies a collection of basic type or enum type, the partialAccessor "<prefix>/<value>" shall identify a member in the collection with the value that corresponds to the URI-decoded value of <value>. Note that this means that a collection shall never contain the same value twice!

It also means that the name from the name column in the Complex type tables shall not be used to address the members in the collection of basic or enum type.

memberAccessor, complex

If the partialAccessor <prefix> identifies a collection of complex type, the partialAccessor "<prefix>/<identifier>" shall identify a member in the collection whose identifier corresponds to <identifier>. If the complex type has an id element, the value of the id element shall be used to uniquely identify the member. For ComplexTypes without an id element, another element shall uniquely identify the member, and in such a case the Type documentation shall specify this element explicitly as the memberAccessor.

valueAccessor

If the partialAccessor <prefix> identifies attribute or element of a basic type or enum type, the partialAccessor "<prefix>/<value>" shall identify a attribute or may be used to check if the attribute has the specified value. This type of accessor is only used for partial retrieve.

Some examples.

To access the permissions attribute in the AR with id myAR (attributeAccessor):
http://m2m.operator.org/accessRights/myAR/permissions.

To access the myPermission element of permissions in this attribute (memberAccessor)
http://m2m.operator.org/accessRights/myAR/permissions/myPermission.

To access the subelement permissionFlags of the above myPermissions subElement (elementAccessor)
http://m2m.operator.org/accessRights/myAR/permissions/myPermission/permissionFlags.

To access the value of READ of the permissionFlag subElement of this permissionHolder (memberAccessor)
http://m2m.operator.org/accessRights/myAR/permissions/myPermission/permissionFlags/READ.

NOTE 1: It is not …/permissionsFlags/flag/READ for the memberAccessor!.

To access the value of myAccessRight of the accessRightID attribute (valueAccessor)
http://m2m.operator.org/applocations/myApplication/accessRightID/myAccessRight.

NOTE 2: This can only be used to check the presence or absence of the value, not for updating, adding or deleting.

A partialAccessor shall have a type that corresponds to the type of the accessor in its last path element in the URI.

An attributeAccessor, elementAccessor or memberAccessor that refers to an attribute or element that has a collection type, is also called a collectionAccessor.

If the prefix does not exist, then any request that addresses an element of that prefix shall be rejected with a STATUS_NOT_FOUND response. This check shall be done after the accessRight permission check if performed, but before any of the other checks mentioned below. In case of subscribe/notify the response is not carried in the response to the creation of the subscription, but in the response embedded in the first (and only) notify.

A representation of a attribute or an element shall be a document with the attribute or element name as its top-level element, and shall conform to the types as defined in clause 11.

10.39.3 Partial Retrieve

Partial retrieve of a resource is done using the <resource>RetrieveRequestIndication. Specifically:

- The primitive type shall be <resource>RetrieveRequestIndication.
- The accessRight check shall be done based on the READ permission.
- All steps are performed in the same order.

However, there are the following modifications:

- The targetID shall include a partialAccessor appended to the resourceURI.
- The used method shall be RETRIEVE.
- The returned result shall correspond to the description below.

General

If any of the prefixes of the partialAccessor are not found a STATUS_NOT_FOUND shall be returned in the response.

Note that this means that it is not possible to distinguish between the resource not being present, the addresses attribute not being present or any of its parents not being present.

We distinguish several cases.

PartialAccessor identifies a value (valueAccessor)

In this case the following shall apply.

If the addressed attribute or element exists in the resource and it has the value as indicated in the valueAccessor, a STATUS_OK (with no content in the response) shall be returned.

If the addressed attribute or element does not exists or if it exists, but does not have the value as indicated in the in the resource a STATUS_NOT_FOUND shall be returned.

PartialAccessor identifies a collection member of basic type or enum type

In this case the following shall apply.

If the addressed member exists in the collection a STATUS_OK (with no content in the response) shall be returned.

If the addressed member does not exist a STATUS_NOT_FOUND shall be returned.

PartialAccesor identifies anything else

In these cases the following shall apply.

If the addressed attribute or element exists in the resource a STATUS_OK shall be returned and the response shall include a representation of the addressed attribute or element.

If the addressed attribute or element does not exist in the resource, a STATUS_NOT_FOUND shall be returned and the response does not include a representation.

10.39.4 Partial retrieve of the content element in a contentInstance resource

The partial retrieval of the *content* element in a *contentInstance* resource is handled in a specific way. In this case the resource representation shall not be the resource representation of the *content* element itself, but the; raw content' as described in clause 11.4, Content, i.e. the response shall directly carry the raw content in its payload.

If the content instance has multiple representations (i.e. its top-level content-type is multipart/alternative) then the content negation procedures from the transport layer apply, i.e. the most desired alternative resource representation is returned.

10.39.5 Partial Update

Partial update of a resource is done using the <resource>UpdateRequestIndication.

Specifically

The primitive type shall be <resource>UpdateRequestIndication:

- The accessRight check shall be done based on the WRITE permission.
- All steps are performed in the same order.

With the following modifications:

- The targetID shall include a partialAccessor appended to the resourceURI.
- The method indicated shall correspond to CREATE, UPDATE or DELETE as show below.
- Validation shall be restricted to the text described below.

- The returned representation response shall correspond to the description below.

General

If the addressed attribute is marked as "NP" in the "M/O in update request" column in clause 9 then the response shall be "STATUS_METHOD_NOT_ALLOWED". Likewise if a element is addressed that is a element of such an 'NP' attribute.

If the valueAccessor is used, the request shall be ignored with a "STATUS_METHOD_NOT_ALLOWED" response.

The clauses below describe how the update is handled with the different methods used in the request.

10.39.5.1 DELETE method (remove an attribute or part of an attribute)

PartialAccessor addresses an attribute

In this case the statusCode as shown in the table below shall be returned. The response shall not include a representation.

Table 10.481: StatusCodes in partial delete response for attributes

Presence in update request	Presence in response	statusCode	Action
M	*any value*	STATUS_METHOD_NOT_ALLOWED	
O	O	STATUS_OK (with no content in the response)	The attribute is removed
O	M* or M	STATUS_OK (with no content in the response)	The attribute reverts to the default value

PartialAccess addresses something else

In these cases the statusCode as shown in table 10.482 shall be returned. The response shall not include a representation.

Table 10.482: StatusCodes in partial delete response for elements

Multiplicity of the addressed element in the parent	statusCode	Action
1	STATUS_METHOD_NOT_ALLOWED	
0..1	STATUS_OK (with no content in the response)	Element is removed from its parent
0..unbounded	STATUS_OK (with no content in the response)	Member is removed from its collection attribute

10.39.5.2 UPDATE method (update an attribute or part of an attribute)

General

In the update, the request shall include the new representation of the attribute or element.

PartialAccessor addresses an attribute

The hosting SCL shall check if the provided value is acceptable according to internal policies.

If the value is accepted, the server shall use the provided value in the resource representation of the updated resource and a STATUS_OK (with no content in the response) shall be returned.

If the provided value is not accepted and the response column is marked M or O then the hosting SCL shall reject the request with a "STATUS_BAD_REQUEST" statusCode.

If the provided value is not acceptable, and the response column is marked M* then the hosting SCL shall update the resource with a value for the attribute that is as close to the requested value as possible, as specified in table 11.36 and a STATUS_OK response code shall be returned together with the modified representation of the attribute.

PartialAccessor addresses a member in a collection, where the member has a basic or enumerated type

If the addressed element is a member in a collection and the member has a basic type or enumeration type, the request shall be rejected with a STATUS_METHOD_NOT_ALLOWED. The proper way to modify members in such a collection is to do a remove and a create.

PartialAccessor addresses a member in a collection, where the member has a complex type

If the addressed element is a member in a collection and the member has a complex type, then the addressed member shall be replaced by the value provided in the request. The identity of the member in the request shall be ignored in this case.

If the addressed member does not exist, the request shall be rejected with a STATUS_NOT_FOUND exception. This means that it is not possible to create a member using an UPDATE. CREATE (add a member to a collection) shall be used instead.

PartialAcessor addresses something else

If a element is addressed, then the element shall be replaced by the new value provided in the UPDATE request, or it shall be created with that value if the attribute did not exist.

If the value of this specific element was modified by the server, then the STATUS_OK response code shall be returned together with the modified representation of the attribute.

If the value was not modified by the server, a STATUS_OK (with no content in the response) shall be returned.

10.39.5.3 CREATE method (add a member to a collection attribute or collection member)

Create is only allowed for collections, if the PartialAccessor addresses an attribute that is not a collection attribute, then the request shall be rejected with a STATUS_METHOD_NOT_ALLOWED response.

If the partialAccessor addresses attribute or element that is a collection that that does not exist, the request shall be rejected with a STATUS_METHOD_NOT_ALLOWED response.

Create a single element

In the create, the request shall include the new representation of the member.

If the member did not yet exist, the response shall be STATUS_CREATED and the collection attribute is extended with the new member. The resourceURI in the response shall be the partial Accessor for the created element.

If the member already exists in the addressed collection, and the member is of a basic or enum type, then the response shall be STATUS_OK (with no content in the response) and the collection attribute is left unchanged.

If the member already exists in the addressed collection, and the member is of a complex type, then the request shall be rejected with a STATUS_CONFLICT statusCode.

Create a multiple element

In the create, the request shall include the new representation of the collection attribute, i.e. a list of multiple member attributes. No resourceURI is included in the response primitive.

If any of the members that are added already exists in the addressed collection, then the response shall be rejected with a STATUS_CONFLICT. If the newly added members do not exists, the collection attribute is extended with the new member, and a STATUS_OK (with no content in the response) is returned.

If all the members in present in the request do not yet exist in the collection (sub-)attribute, the response shall be STATUS_OK (with no content in the response) and the members are added to the representation of the collection (sub-)attribute.

If none or some of the members present in the request already exists in the addressed collection attribute or element and the members are of a basic or enum type, then the response shall be STATUS_OK (with no content in the response) and the collection attribute is extended with the any members in the request that are not yet present in the collection (sub-)attribute.

If any of the member present in the request already exists in the addressed collection (sub-)attribute, and the members are of a complex type, then the request shall be rejected with a STATUS_CONFLICT statusCode.

10.39.6 Partial Subscribe and Partial Notify

Partial subscriptions of a resource shall use the subscriptionCreateRequestIndication addressing the subscriptions collection child of the subscribed-to resource. The filterCriteria of the subscription contain the partial accessor.

If the subscription was successful and the resource is modified, the hosting SCL shall check if the modifications of the resource influence those part of the resource that match the part selected partialAccessor. If so, the hosting SCL shall send a notification. However, the notification shall only contain that part of the resource selected by the partialAccessor, just as described in clause 10.39.3.

If at the time of subscription, a partial retrieve with the same filterCriteria would result in an error response because of any of the specific errors mentioned in clause 10.39.3, the creation of the subscription shall be accepted, but the subscription is immediately terminated, and a notify containing the error response indicated in the partial retrieve (clause 10.39.3) is returned in the notify. Similar, if in the subscribed-to resource is modified in such a way that a partial retrieve with the attributeAccessor as specified in the filterCriteria would return an error, a final notify is send as specified in clause 10.25.7.3 and the subscription is terminated.

11 Data types and attributes for resources and messages

11.1 Assumptions

The data types described in the present clause are based on sequences of **characters**, referred as "**characters**" in the following data type descriptions. Different character encodings are supported. The character-encoding may be UTF-8 or UTF-16.

The **minimum** size of data types is of one character.

The **maximum** size of data types is not specified in some cases for not limiting the representational capability, e.g. for an URL.

The attributes described in this clause shall apply to both the Resource data models and the methods on mIa, dIa, and mId interfaces/ reference points.

11.2 Basic data types

The M2M system shall support Resources and mIa, dIa and mId reference points according to the following basic data types derived by XSD definitions defined in annex B.

The type and values shall be supported according to the description given in table 11.1.

Table 11.1: Basic data types

TypeName	xsd type	Examples	Description
Long	xsd:long	Examples: "*0*", "*-123*", "*+10*", "*9223372036854775807*", "*-9223372036854775808*"	Sequence of numeric characters that is interpreted as an signed integer decimal number with 64-bits precision. An optional leading sign is allowed. If the sign is omitted, "+" is assumed
String	xsd:string	Example: "*Hello World!*"	Sequence of alphanumeric characters that is interpreted as it is without conversions
DateTime	xsd:dateTime	Example: "*2002-10-10T17:00:00.000+05:00*"	Structured alphanumeric data which is an extension of ISO 8601 [1]

TypeName	xsd type	Examples	Description
AnyURI	xsd:anyURI	Example: "*http://www.telecomitalia.it:8080/m2m/scls/scl1/applications?searchString=tag1&searchString=tag2*" or "*platform1.service_provider1.m2m_service.network_operator/resource*"	Structured alphanumeric data type as inspired by RFC 3986 [23]. URIs shall be either be absolute URIs or relative URIs
AnyType	xsd:anyType	Example: <this>can be anything</this>	Any type
Token	xsd:NMTOKEN	Example: "*some*", "*some_string_without_spaces*"	A token is a string that does not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters, or any other whitespace
Stoken	xsd:token	Example "My Access Network"	A Stoken is a string defined in the same way as a Token, with the exception that single spaces are allowed within the string.
MemorySize	xsd:NMTOKEN restricted by regular expression "[0-9]{1,15}[BKMGT]"	Examples: "1432B" "100K" "16M".	A Token that consists of an integer expression plus a memory unit out of B (Bytes), K (1 024 Bytes), M (1 048 576 Bytes), G (1 073 741 824 Bytes), or T (1099511627776 Bytes).
Duration	xsd:duration	Examples: "*PT10.123S*", "*P2Y2M3DTH10M1S*", "*-PT1S*"	Duration represents a duration of time. The value space of duration is a six-dimensional space where the coordinates designate the Gregorian year, month, day, hour, minute, and second components defined in § 5.5.3.2 of ISO 8601 [1], respectively. These components are ordered in their significance by their order of appearance i.e. as year, month, day, hour, minute, and second. An optional preceding minus sign ('-') is allowed, to indicate a negative duration. The seconds part may·have a decimal fraction
ScheduleString	xsd:string	Examples: *"* 0-6 * * 1-6" would mean the period from 0:00h to 6:00h (inclusive) on any week day (Monday through Friday)* *"* 22-23,0-4 * * 1-6" would mean the period from 22:00h to 04:00h on any week day (Monday through Friday)*	String formatted according to the CRONTAB input file definition in ISO/IEC/IEEE 9945 [44] The schedule that is defined by this string is the sum of all time spans (in one minute granularity) that match with the given CRONTAB string Note the command field of a CRONTAB line is not used in this definition.
UnsignedLong	xsd:unsignedLong	Examples: "*0*", "*+10*", "00*9223372036854775807*", *"0",* *"+0000000000000000000005"* *"1"*	The value space is the range of integers between 0 and 18446744073709551615 - the unsigned values that can fit in a word of 64 bits. Its lexical space allows an optional + sign and leading zeros before the significant digits
UnsignedInt	xsd:unsignedInt	Examples: "*4294967295",* *"0",* *"+0000000000000000000005"* *"1"*	The value space of xsd:unsignedInt is the range of integers between 0 and 4294967295 - the unsigned values that can fit in a word of 32 bits. Its lexical space allows an optional + sign and leading zeros before the significant digits
HexBinary	Xsd:hexBinary	Examples: *"3f3c6d78206c657673726f693d6e3122302e20226e656f636964676e223d54552d4622383e3f"*	The value space of xsd:hexBinary is the set of all binary contents; its lexical space is a simple coding of each octet as its hexadecimal value

11.3 Enumeration types

The M2M system shall support the following enumeration data types.

Table 11.2: Enumeration types

TypeName	Values	Description
CmdType	Predefined values: "RESET" "REBOOT" "UPLOAD" "DOWNLOAD" "SCHEDULEDOWNLOAD" "SCHEDULEINFORM" "SCHEDULEDUCHANGE"	Enumeration of the type of management commands
ChannelType	Predefined values: "LONG_POLLING"	Enumeration of the type of the notificationChannel. Currently only long polling is supported as a type
ExecStatus	Predefined values: "INITIATED" "STARTED" "FINISHED" "CANCELLING"	Enumeration of the status of an command execution instance
LocationContainerType	Predefined values: "LOCATION_SERVER_BASED" "APPLICATION_GENERATED"	Enumeration of the source of location information
MemberType	Predefined values: "APPLICATION": application resource "CONTAINER": container resource "ACCESS_RIGHT": access rights resource "SCL"; SCL resource "SCL_BASE": sclBase resource of a service capability layer "LOCATION_CONTAINER": location container resource "MGMT_OBJ": mgmtObj resource "MGMT_CMD": mgmtCmd resource "ATTACHED_DEVICE": attachedDevice resource "MIXED": mixed resource types	Enumeration of types of group members or subgroup members
OnlineStatus	Predefined values: "ONLINE": scl registered and able to communicate "OFFLINE": scl registered but unable to communicate "NOT_REACHABLE": scl registered but unreachable for the moment	Enumeration of types of status of an scl stored in an <*scl*> resource The *onlineStatus* of the <*scl*> resource is ONLINE if there is at least one *m2mPoc* resource in its collection with an *onlineStatus* set to ONLINE or if there is long polling active for the <*scl*> as indicated in TS 102 690 [2] Resource <scl> The *onlineStatus* of the <*scl*> resource is NOT_REACHABLE if all the *m2mPoc* resources in its collection have an *onlineStatus* set to NOT_REACHABLE The *onlineStatus* of the <*scl*> resource is OFFLINE in all other cases (i.e. if there are no *m2mPoc* resources in the collection or if all m2mPoc resource in the collection have their *onlineStatus* attribute set to OFFLINE)
PermissionFlag	Predefined values: "*READ*": read enabled "*WRITE*": write enabled "*DELETE*": deletion enabled "*CREATE*": create child enabled "*DISCOVER*": discovery enabled	Enumeration of permission flag options

TypeName	Values	Description
RcatType	Predefined values: "RCAT_0" "RCAT_1" "RCAT_2" "RCAT_3" "RCAT_4" "RCAT_5" "RCAT_6" "RCAT_7"	Enumeration of the different possible request categories that SAF handling can process. Interpretation of what a specific RCAT value means is up to agreement between M2M SP and access network provider(s)
StatusCode	Predefined values: "STATUS_OK" "STATUS_CREATED" "STATUS_ACCEPTED" "STATUS_BAD_REQUEST" "STATUS_PERMISSION_DENIED" "STATUS_FORBIDDEN" "STATUS_NOT_FOUND" "STATUS_METHOD_NOT_ALLOWED" "STATUS_NOT_ACCEPTABLE" "STATUS_REQUEST_TIMEOUT" "STATUS_CONFLICT" "STATUS_UNSUPPORTED_MEDIA_TYPE" "STATUS_INTERNAL_SERVER_ERROR" "STATUS_NOT_IMPLEMENTED" "STATUS_BAD_GATEWAY" "STATUS_SERVICE_UNAVAILABLE" "STATUS_GATEWAY_TIMEOUT" "STATUS_DELETED" "STATUS_EXPIRED"	Enumeration of status codes applied to procedures invoked by primitives
SubscriptionType	Predefined values: "*ASYNCHRONOUS*" : "*SYNCHRONOUS*" :	Enumeration of the type of subscriptions
MgmtProtocolType	Predefined values: "OMA_DM" "BBF_TR069"	Enumeration of the type of management protocols

TypeName	Values	Description
PrimitiveType	Predefined values: "SCLBASE_RETRIEVE_REQUEST" "SCLBASE_RETRIEVE_RESPONSE" "SCLBASE_UPDATE_REQUEST" "SCLBASE_UDPATE_RESPONSE" "SCLS_RETRIEVE_REQUEST" "SCLS_RETRIEVE_RESPONSE" "SCLS_UPDATE_REQUEST" "SCLS_UDPATE_RESPONSE" "SCL_CREATE_REQUEST" "SCL_CREATE_RESPONSE" "SCL_RETRIEVE_REQUEST" "SCL_RETRIEVE_RESPONSE" "SCL_UPDATE_REQUEST" "SCL_UDPATE_RESPONSE" "SCL_DELETE_REQUEST" "SCL_DELETE_RESPONSE" "APPLICATIONS_RETRIEVE_REQUEST" "APPLICATIONS_RETRIEVE_RESPONSE" "APPLICATIONS_UPDATE_REQUEST" "APPLICATIONS_UDPATE_RESPONSE" "APPLICATION_CREATE_REQUEST" "APPLICATION_CREATE_RESPONSE" "APPLICATION_RETRIEVE_REQUEST" "APPLICATION_RETRIEVE_RESPONSE" "APPLICATION_UPDATE_REQUEST" "APPLICATION_UDPATE_RESPONSE" "APPLICATION_DELETE_REQUEST" "APPLICATION_DELETE_RESPONSE" "APPLICATION_ANNC_CREATE_REQUEST" "APPLICATION_ANNC_CREATE_RESPONSE" "APPLICATION_ANNC_RETRIEVE_REQUEST" "APPLICATION_ANNC_RETRIEVE_RESPONSE" "APPLICATION_ANNC_UPDATE_REQUEST" "APPLICATION_ANNC_UDPATE_RESPONSE" "APPLICATION_ANNC_DELETE_REQUEST" "APPLICATION_ANNC_DELETE_RESPONSE" "ACCESS_RIGHTS_RETRIEVE_REQUEST" "ACCESS_RIGHTS_RETRIEVE_RESPONSE" "ACCESS_RIGHTS_UPDATE_REQUEST" "ACCESS_RIGHTS_UDPATE_RESPONSE" "ACCESS_RIGHT_CREATE_REQUEST" "ACCESS_RIGHT_CREATE_RESPONSE" "ACCESS_RIGHT_RETRIEVE_REQUEST" "ACCESS_RIGHT_RETRIEVE_RESPONSE" "ACCESS_RIGHT_UPDATE_REQUEST" "ACCESS_RIGHT_UDPATE_RESPONSE" "ACCESS_RIGHT_DELETE_REQUEST" "ACCESS_RIGHT_DELETE_RESPONSE" "ACCESS_RIGHT_ANNC_CREATE_REQUEST" "ACCESS_RIGHT_ANNC_CREATE_RESPONSE" "ACCESS_RIGHT_ANNC_RETRIEVE_REQUEST" "ACCESS_RIGHT_ANNC_RETRIEVE_RESPONSE" "ACCESS_RIGHT_ANNC_UPDATE_REQUEST" "ACCESS_RIGHT_ANNC_UDPATE_RESPONSE" "ACCESS_RIGHT_ANNC_DELETE_REQUEST" "ACCESS_RIGHT_ANNC_DELETE_RESPONSE" "CONTAINERS_RETRIEVE_REQUEST" "CONTAINERS_RETRIEVE_RESPONSE" "CONTAINERS_UPDATE_REQUEST" "CONTAINERS_UDPATE_RESPONSE" "CONTAINER_CREATE_REQUEST" "CONTAINER_CREATE_RESPONSE" "CONTAINER_RETRIEVE_REQUEST" "CONTAINER_RETRIEVE_RESPONSE" "CONTAINER_UPDATE_REQUEST" "CONTAINER_UDPATE_RESPONSE"	Predefined values

TypeName	Values	Description
	"CONTAINER_DELETE_REQUEST" "CONTAINER_DELETE_RESPONSECONTAINER_"ANNC_CREATE_REQUEST" "CONTAINER_ANNC_CREATE_RESPONSE" "CONTAINER_ANNC_RETRIEVE_REQUEST" "CONTAINER_ANNC_RETRIEVE_RESPONSE" "CONTAINER_ANNC_UPDATE_REQUEST" "CONTAINER_ANNC_UDPATE_RESPONSE" "CONTAINER_ANNC_DELETE_REQUEST" "CONTAINER_ANNC_DELETE_RESPONSE" "LOCATION_CONTAINER_CREATE_REQUEST" "LOCATION_CONTAINER_CREATE_RESPONSE" "LOCATION_CONTAINER_RETRIEVE_REQUEST" "LOCATION_CONTAINER_RETRIEVE_RESPONSE" "LOCATION_CONTAINER_UPDATE_REQUEST" "LOCATION_CONTAINER_UDPATE_RESPONSE" "LOCATION_CONTAINER_DELETE_REQUEST" "LOCATION_CONTAINER_DELETE_RESPONSE" "LOCATION_CONTAINER_ANNC_CREATE_REQUEST" "LOCATION_CONTAINER_ANNC_CREATE_RESPONSE" "LOCATION_CONTAINER_ANNC_RETRIEVE_REQUEST" "LOCATION_CONTAINER_ANNC_RETRIEVE_RESPONSE "LOCATION_CONTAINER_ANNC_UPDATE_REQUEST" "LOCATION_CONTAINER_ANNC_UDPATE_RESPONSE" LOCATION_CONTAINER_ANNC_DELETE_REQUEST" "LOCATION_CONTAINER_ANNC_DELETE_RESPONSE" "CONTENT_INSTANCES_RETRIEVE_REQUEST" "CONTENT_INSTANCES_RETRIEVE_RESPONSE" "CONTENT_INSTANCE_CREATE_REQUEST" "CONTENT_INSTANCE_CREATE_RESPONSE" "CONTENT_INSTANCE_RETRIEVE_REQUEST" "CONTENT_INSTANCE_RETRIEVE_RESPONSE" "CONTENT_INSTANCE_DELETE_REQUEST" "CONTENT_INSTANCE_DELETE_RESPONSE" "GROUPS_RETRIEVE_REQUEST" "GROUPS_RETRIEVE_RESPONSE" "GROUPS_UPDATE_REQUEST" "GROUPS_UDPATE_RESPONSE" "GROUP_CREATE_REQUEST" "GROUP_CREATE_RESPONSE" "GROUP_RETRIEVE_REQUEST" "GROUP_RETRIEVE_RESPONSE" "GROUP_UPDATE_REQUEST" "GROUP_UDPATE_RESPONSE" "GROUP_DELETE_REQUEST" "GROUP_DELETE_RESPONSE" "GROUP_ANNC_CREATE_REQUEST" "GROUP_ANNC_CREATE_RESPONSE" "GROUP_ANNC_RETRIEVE_REQUEST" "GROUP_ANNC_RETRIEVE_RESPONSE" "GROUP_ANNC_UPDATE_REQUEST" "GROUP_ANNC_UDPATE_RESPONSE" "GROUP_ANNC_DELETE_REQUEST" "GROUP_ANNC_DELETE_RESPONSE" "MEMBERS_CONTENT_CREATE_REQUEST" "MEMBERS_CONTENT_CREATE_RESPONSE" "MEMBERS_CONTENT_RETRIEVE_REQUEST" "MEMBERS_CONTENT_RETRIEVE_RESPONSE" "MEMBERS_CONTENT_UPDATE_REQUEST" "MEMBERS_CONTENT_UDPATE_RESPONSE" "MEMBERS_CONTENT_DELETE_REQUEST" "MEMBERS_CONTENT_DELETE_RESPONSE"	

TypeName	Values	Description
	"SUBSCRIPTIONS_RETRIEVE_REQUEST" "SUBSCRIPTIONS_RETRIEVE_RESPONSE" "SUBSCRIPTION_CREATE_REQUEST" "SUBSCRIPTION_CREATE_RESPONSE" "SUBSCRIPTION_RETRIEVE_REQUEST" "SUBSCRIPTION_RETRIEVE_RESPONSE" "SUBSCRIPTION_UPDATE_REQUEST" "SUBSCRIPTION_UDPATE_RESPONSE" "SUBSCRIPTION_DELETE_REQUEST" "SUBSCRIPTION_DELETE_RESPONSE" "SUBSCRIPTION_NOTIFY_REQUEST" "SUBSCRIPTION_NOTIFY_RESPONSE" "M2MPOCS_RETRIEVE_REQUEST" "M2MPOCS _RETRIEVE_RESPONSE" "M2MPOC_CREATE_REQUEST" "M2MPOC_CREATE_RESPONSE" "M2MPOC_RETRIEVE_REQUEST" "M2MPOC_RETRIEVE_RESPONSE "M2MPOC_UPDATE_REQUEST" "M2MPOC_UDPATE_RESPONSE" "M2MPOC_DELETE_REQUEST" "M2MPOC_DELETE_RESPONSE" "MGMTOBJS_RETRIEVE_REQUEST" "MGMTOBJS_RETRIEVE_RESPONSE" "MGMTOBJS_UPDATE_REQUEST" "MGMTOBJS_UDPATE_RESPONSE" "MGMTOBJ_CREATE_REQUEST" "MGMTOBJ_CREATE_RESPONSE" "MGMTOBJ_RETRIEVE_REQUEST" "MGMTOBJ_RETRIEVE_RESPONSE" "MGMTOBJ_UPDATE_REQUEST" "MGMTOBJ_UDPATE_RESPONSE" "MGMTOBJ_DELETE_REQUEST" "MGMTOBJ_DELETE_RESPONSE" "MGMTOBJ_EXECUTE_REQUEST" "MGMTOBJ_EXECUTE_RESPONSE" "PARAMETERS_CREATE_REQUEST" "PARAMETERS_CREATE_RESPONSE" "PARAMETERS_RETRIEVE_REQUEST" "PARAMETERS_RETRIEVE_RESPONSE" "PARAMETERS_UPDATE_REQUEST" "PARAMETERS_UDPATE_RESPONSE" "PARAMETERS_DELETE_REQUEST" "PARAMETERS_DELETE_RESPONSE" "PARAMETERS_EXECUTE_REQUEST" "PARAMETERS_EXECUTE_RESPONSE" "MGMTCMD_CREATE_REQUEST" "MGMTCMD_CREATE_RESPONSE" "MGMTCMD_RETRIEVE_REQUEST" "MGMTCMD_RETRIEVE_RESPONSE" "MGMTCMD_UPDATE_REQUEST" "MGMTCMD_UDPATE_RESPONSE" "MGMTCMD_DELETE_REQUEST" "MGMTCMD_DELETE_RESPONSE" "MGMTCMD_EXECUTE_REQUEST" "MGMTCMD_EXECUTE_RESPONSE" "EXECINSTANCES_RETRIEVE_REQUEST" "EXECINSTANCES _RETRIEVE_RESPONSE" "EXECINSTANCE_RETRIEVE_REQUEST" "EXECINSTANCE_RETRIEVE_RESPONSE" "EXECINSTANCE_DELETE_REQUEST" "EXECINSTANCE_DELETE_RESPONSE" "EXECINSTANCE_EXECUTE_REQUEST" "EXECINSTANCE_EXECUTE_RESPONSE" "ATTACHEDDEVICES_RETRIEVE_REQUEST" "ATTACHEDDEVICES_RETRIEVE_RESPONSE" "ATTACHEDDEVICES_UPDATE_REQUEST"	

TypeName	Values	Description
	"ATTACHEDDEVICES_UDPATE_RESPONSE" "ATTACHEDDEVICE_CREATE_REQUEST" "ATTACHEDDEVICE_CREATE_RESPONSE" "ATTACHEDDEVICE_RETRIEVE_REQUEST" "ATTACHEDDEVICE_RETRIEVE_RESPONSE" "ATTACHEDDEVICE_UPDATE_REQUEST" "ATTACHEDDEVICE_UDPATE_RESPONSE" "ATTACHEDDEVICE_DELETE_REQUEST" "ATTACHEDDEVICE_DELETE_RESPONSE" "NOTIFICATION_CHANNELS_RETRIEVE_REQUEST" "NOTIFICATION_CHANNELS_RETRIEVE_RESPONSE" "NOTIFICATION_CHANNEL_CREATE_REQUEST" "NOTIFICATION_CHANNEL_CREATE_RESPONSE" "NOTIFICATION_CHANNEL_RETRIEVE_REQUEST" "NOTIFICATION_CHANNEL_RETRIEVE_RESPONSE" "NOTIFICATION_CHANNEL_DELETE_REQUEST" "NOTIFICATION_CHANNEL_DELETE_RESPONSE" "NOTIFICATION_CHANNEL_NOTIFY_REQUEST" "NOTIFICATION_CHANNEL_NOTIFY_RESPONSE" "DISCOVERY_RETRIEVE_REQUEST" "DISCOVERY_RETRIEVE_RESPONSE"	
APocHandling	Predefined values: "SHALLOW" "DEEP"	Possible ways of doing aPoc retargeting. SHALLOW means that only exact or shallow prefix matches (1 level deep) to elements in the aPoCPaths are retargeted, DEEP means that any prefix match will result in retargeting. See clause 10.3.2.23 for details
ConsistencyStrategy	Predefined values: "ABANDON_GROUP" "ABANDON_MEMBER" "MODIFY_TYPE"	Enumeration of the type of consistency strategies.

11.4 Complex data types

The M2M system shall support the following complex data types derived by data types as described in clause 2.3.

Content

Content of a content instance. This shall be coded in two ways.

- as a base64Binary with a contentType attribute:
  - This is used in the canonical XML representation of the contentInstance resource.
  - This can be mapped to a MIME RFC 2045 [14], RFC 2046 [15], RFC 2047 [16] part with the same content-type using the MTOM/XOP rules as defined in annex C.
  - This format is the format used when the content is embedded in a contentInstance resource representation.

Table 11.3: Content type

Name	Type	Description
xmlmime:contentType	String	The content type of the content that is provided in the value This is mapped to an xml attribute
-	base64Binary	The actual content as a base64 encoded binary. This is mapped to the data part of the content

- as an entity or payload on the transport layer. In this case the content-Type is transported as meta-data in the transport layer as well, e.g. as a content-type header in HTTP or as a content-type code in CoAP. This format is also called the 'raw content':
  - This format shall used in responses when the *content* member of the <*contentInstance*> resource is retrieved.
  - This format may be used in contentInstanceCreateRequestIndications, in case no contentInstance specific meta-data is provided (e.g. in case no delayTolerance element is to be provided in the *contentInstance*)/.

For example, a contentInstanceCreateRequestIndication could be mapped as follows:

```
POST base/containers/myContainer/contentInstances
Host: m2m.operator.org
Content-Type: image/jpeg
Transfer-Encoding: binary
Content-Length: 42

… binary content of the picture…
```

If the contentInstance contains a multipart/alternative content, then this shall regarded as the content having multiple (alternative) representations. Normal content negotiation mechanism then apply when accessing the raw content of a contentInstance that stores multiple representations.

ContentType

Table 11.4: ContentType type

Name	Type	Predefined values and examples	Description
contentType	String[0..unbounded]		A contentType of a *contentInstance* resource

ErrorInfo

Type for error related information. Including the statusCode and additionalInformation.

Table 11.5: ErrorInfo type

Name	Type	Description
statusCode	StatusCode	The error statusCode. This means that STATUS_OK or STATUS_CREATED is not allowed.
additionalInfo	String[0..1]	Additional error information. This optional element gives the detailed reason for the error. This is useful for debugging purposes.

NamedReferenceCollection

List of references to named resources.

Table 11.6: NamedReferenceCollection type

Name	Type	Description
namedReference	ReferenceToNamedResource [0..unbounded]	Reference to named resource

ReferenceToNamedResource

Reference to resource with optional id.

Table 11.7: ReferenceToNamedResource type

Name	Type	Description
id	Token[0..1]	Id of the referenced resource for resources that are local children
reference	AnyURI	Reference to the resource

ChannelData

Abstract type used of deriving specific channel data.

LongPollingChannelData

Derived type from ChannelData.

Table 11.8

Name	Type	Description
longPollingURI	AnyURI	The URI to be used for long polling requests

ContactInfo

List of content instances.

Table 11.9

Name	Type	mode	Description
contactURI	AnyURI [0..1]	CHOICE	Only the schema, the host and the port information are used. This shall be either an IP-address or a fully qualified domain name that still needs to be converted.
other	Any[0..1]	CHOICE	Unspecified information. This may contain other information.

ContentInstanceCollection

List of content instances.

Table 11.10

Name	Type	Description
contentInstance	ContentInstance [0..unbounded]	List of contentInstance resource representations.

AnyURIList

Dynamic list of URIs.

Table 11.11

Name	Type	Description
reference	AnyURI[0..unbounded]	A reference to a REST resource as an absolute or relative URI

APocPaths

Dynamic list of tokens representing searchstrings.

Table 11.12

Name	Type	Description
aPocPath	APocPath[0..unbounded]	List of application PoCs

APocPath

Dynamic list of tokens representing searchstrings.

Table 11.13

Name	Type	Description
path	AnyURI	The path. The uriEncoded value of this element shall be used as the memberAccessor for this collection.
accessRightID	AnyURI[0..1]	The optional accessRight associated with the path.
searchStrings	SearchStrings[0..1]	The optional searchStrings associated with the path.

SearchStrings

Dynamic list of tokens representing searchstrings.

Table 11.14

Name	Type	Description
searchString	Token[0..unbounded]	A search string token.

Permissions

Set of Permissions. Each permission in the list shall have an unique ID that can used to identify the specific permission.

Table 11.15

Name	Type	Description
permission	Permission [0..unbounded]	The type of permission

Permission

A Permission specified who (permissionHolders) is allowed to do what (permissionFlags).

Table 11.16

Name	Type	Description
permissionFlags	PermissionFlags	List of enabled permissions.
permissionHolders	PermissionHolders	Specifies the entities that have certain accessRights specified by the corresponding permissionFlags.
id	String	Identification of the permission. (typically mapped to an xml-attribute).

PermissionFlags

Set of Permission flags. Note that each flag shall appear only once.

Table 11.17

Name	Type	Description
flag	PermissionFlag[0..unbounded]	The type of permission

PermissionHolders

Table 11.18

Name	Type	Predefined values and examples	Description
holderRefs	HolderRefs[0..1]	<holderRefs><holderRef> http://domain1.myoperator.com/m2m/applications/app1</holderRef></holderRefs>>	Reference to an application, sclBase, scl or group resource.
applicationIDs	AppliationIDs[0..1]		ApplicationIDs of applications that are permission holders, regardless of the URI of the related application resource.
sclIDs	SCLIDs[0..1]		SCLIDs of the scls that are permissionHolders.
all	emptyType [0..1]	< all/>	Any application or SCL that is authenticated.
domains	Domains[0..1]	<domains><domain>http://domain1.myoperator.com/m2m/</domain> <domain>http://domain2.myoperator.com/m2m/,</domain></domains>	Any application, scl, sclBase that starts with the specified URI.

ApplicationIDs

Table 11.19

Name	Type	Predefined values and examples	Description
applicationID	AnyURI[0..unbounded]	<applicationID>uri:12334567</applicationID>	ID of an application. This supports the use of non-URI for nomadic applications (see note).
NOTE: The applicationID is globally unique.			

SCLIDs

Table 11.20

Name	Type	Predefined values and examples	Description
sclID	AnyURI[0..unbounded]	<sclID>uri:1234567</sclID>	ID of an SCL. Including the ID of the SCL as permissionHolder has the same effect as including <remoteSCLBase>/scls/<SCLID> as a permissionHolder.

HolderRefs

Table 11.21

Name	Type	Predefined values and examples	Description
holderRef	AnyURI[0..unbounded]	<holderRef> http://domain1.myoperator.com/m2m/applications/app1</holderRef>	Reference to an application, scl, sclBase or group resource. The group resource shall have a memberType of "APPLICATION" or "SERVICE_CAPABILITY_LAYER" or "SCL".

Domains

Table 11.22

Name	Type	Predefined values and examples	Description
domain	AnyURI[0..unbounded]	<domain> http://domain1.myoperator.com/m2m</domain>	Any application, scl or sclBase that starts with the specified URI.

FilterCriteria

FilterCriteria are used in several situations:

1) During RETRIEVE (for discovery and the contentInstances resource).

2) During subscribe/notify. Subscription to contentInstances is treated as a special case, since this is a combination of using the criteria for retrieve and notify.

FilterCriteria serve two purposes:

- Matching. This is used Only matching resources in a the solution space are to be returned in a retrieve response or are send in a notify request.
- Transforming; This is used to select which attributes of a resource representation are returned in a RETRIEVE response or in a notify request.

The filterCriteria shall be applied as follows:

- RETRIEVE with filterCriteria:
  - The hosting SCL matches every resource in the solution space against all the match criteria that are present in the filterCriteria. If a resource matches, the transforming filters are applied and the resulting resource representation is included in the result set. The result set may be empty.
  - The solution space is defined as follows:
    - For the discovery resource; the set of resources that hierarchically reside under the search root defined by the *searchPrefix* attribute.
    - For a contentInstances resource; the set of contentInstance resources in the collection.
- SUBSCRIBE & initial notify:
  - The hosting SCL matches every resource in the solution space against all the match criteria that are present in the filterCriteria. If a resource matches, the transforming filters are applied and the resulting resource representation is included in the result set.
  - If the result set is empty, no initial notify shall be sent.
  - The solution space is the defined with respect to the subscribe-to resource as follows:
    - For a contentInstances resource, the solution space is the set of contentInstance resources in the collection.
    - For other subscribable resources; the subscribed-to resource itself.
- SUBSCRIBE & subsequent notify:
  - The hosting SCL matches every resource in the solution space against all the match criteria that are present in the filterCriteria. If a resource matches, the transforming filters are applied and the resulting resource representation is included in the result set.
  - If the result set is empty, no subsequent notify shall be sent.
  - If the hosting SCL has the capability to detect that the notify would have the same value as the previous notify send for the same subscription, no notify shall be sent.

- The solution space is the defined with respect to the subscribe-to resource as follows:
  - For a contentInstances resource; the set of contentInstance resources that have been **added** to the collection since the last notify was send for this subscription.
  - For other subscribable resources; the subscribed-to resource itself.

Table 11.23

Name	Type	Predefined values and examples	Description
ifModifiedSince	DateTime[0..1]		A resource matches this criterion if and only if the lastModified attribute of the resource is chronologically after the specified value. It is specifically useful to suppress the initial notify when used in a create subscription (see clause 10.25.2).
ifUnmodifiedSince	DateTime[0..1]		A resource matches this criterion if and only if the lastModifiedAttribute of the resource is chronologically before the specified value.
ifNoneMatch	String[0..unbounded]		A resource matches this criterion if the etag of the resource does NOT match any of the specified values. This is specifically useful to suppress the initial notify when used in a create subscription (see clause 10.25.2).
attributeAccessor	AnyURI		Relative URI indicating the attribute or element in the resource. This is equivalent to including the similar attributeAccessor in a partial addressing retrieve operation, i.e. only the specified attribute or element value is represented in the response or equivalent notify.
searchString	String[0..unbounded]		A resource matches this criterion if the resource has a searchStrings attribute and one of the searchString values in the searchStrings attribute is the same as the specified value. If multiple searchStrings are specified, the logical operation is AND, i.e. only resources that contain all the specified searchStrings match the criterion.
createdAfter	DateTime[0..1]		A resource matches this criterion if and only if the creationTime attribute of the resource is chronologically after the specified value.
createdBefore	DateTime[0..1]		A resource matches this criterion if and only if the creationTime attribute of the resource is chronologically before the specified value.

ContentInstancesFilterCriteria extends FilterCriteria

This is the filterCriteria specifically used for retrieving a *contentInstances* resource, for subscribing to a *contentInstances* resource, or for subscribing to the latest or oldest URIs in a *contentInstances* collection.

Table 11.24

Name	Type	Predefined values and examples	Description
sizeFrom	Long[0..1]		A contentInstance resource matches this criterion if and only if the contentSize attribute of the resource is greater than the specified value.
sizeUntil	Long[0..1]		A contentInstance resource matches this criterion if and only if the contentSize attribute of the resource is smaller than the specified value.
contentType	String[0..unbounded]		A contentInstance resource matches this criterion if any of the values in the contentTypes attribute of the resource matches the specified value The specified value may include wildcards. E.g. image/* matches all images. If a *contentInstance* resource matches, only the matching representations shall be returned. If more than one of its alternative representation matches then multiple representations are returned inside a multipart/alternative content-type.
metaDataOnly	Boolean[0..1]		Return only meta-data and not the content of contentInstances. If not specified this defaults to FALSE, i.e. the content is returned as well.

The logical operation in case of filtering on multiple criteria is AND.

IValResult

Reported Integrity Validation results.

Table 11.25

Name	Type	Description
ivalResults	Long	The raw integrity validation results binary list of "Pass (1) or Fail(0)" for component to functionality mapping defined by M2M SP.
signedIvalResult	Long	The signed integrity validation results.
secureTimeStamp	DateTime	Secure timestamp of when integrity validation was reported.

Schedule

Type for defining complex possibly periodic schedules.

Table 11.26

Name	Type	Description
schedule	ScheduleString [0..unbounded]	List of ScheduleStrings defining a schedule that is the sum of all time spans defined by each ScheduleString.

TrpdtType

Type for setting the Tolerable Request Processing Delay Time (TRPDT) in SAF handling.

Table 11.27

Name	Type	Description
tolerableDelay	Duration[0..1]	The duration that is tolerable for the processing of an issued request that is pending in SAF processing.
tolerableTime	Time[1 - numberOfOccurrence('TolerableDelay')]	The absolute time until which a request may be pending in SAF handling. This field of the TrpdtType may only be populated with a value when no 'TolerableDelay' field is used.

AnnounceTo

Table 11.28

Name	Type	Predefined values and examples	Description
activated	Boolean[0..1]	TRUE, FALSE	Used to set the activation status of an announcement to TRUE or FALSE. This shall not be used to de-announce, i.e. once this value is set to TRUE, it can no longer be set back to FALSE. If not present, then the Issuer SCL will treat activation status as TRUE.
sclList	AnyURIList		The list of sclBase URIs of the SCLs where the resource has to be announced (in requests) or is announced (in responses).
global	Boolean[0..1]	TRUE, FALSE	Indicates whether the announcements are applicable to *all* resource created by the issuer that do not have an explicit announceTo attribute, at the creation time of that resource. This attribute is ONLY applicable in the announceTo attribute of the application registration resource. Its value is ignored in any other resource. Changes in this or any other values of the announceTo in the application resource's announceTo do not affect already created resources. If not present, this treated the same as if it would be set to FALSE.

AnyURIList

Table 11.29

Name	Type	Predefined values and examples	Description
reference	AnyURI[0..unbounded]		The URI

ExecReqArgsList

The list of command request arguments.

Table 11.30

Name	Type	Description
execReqArg	ExecReqArg[0..unbounded]	List of command request arguments

ExecReqArg
A single command request argument.

Table 11.31

Name	Type	Description
name	String	The name of the command request argument
value	AnyType	The value of the command request argument

ExecResultList

The list of command request arguments.

Table 11.32

Name	Type	Description
execResultItem	ExecResultItem[0..unbounded]	List of command results

ExecResultItem

A result item.

Table 11.33

Name	Type	Description
name	String	The name of the result item
value	AnyType	The value of the result item

MembersContentResponses

Table 11.34

Name	Type	Description
status	MembersContentResponse[0..unbounded]	The status of an individual response

MembersContentResponse

Table 11.35

Name	Type	Description
id	AnyURI	The URI of the members resource that initiated the response.
statusCode	StatusCode	The status code returned by the member resource identified by the id attribute.
eTag	Token[0..1]	The etag returned in the response from the member resource identified by the id attribute.
resourceURI	AnyURI	In case that the member resource was created successfully, this shall be resourceURI obtained from the location header field. This is an optional attribute only applied to the create operation.
lastModifiedTime	DateTime	The lastModifiedTime returned in the response form the member resource identified by the id attribute.
resultBody	Content	The result returned in the response by the member identified by the id attribute. Its type shall be base64Binary. For member resource this shall either be the resource representation of the member resource if existed in case the statusCode attribute includes a successful status code and a representation was returned in the response. Or it shall be the errorInfo representation in case the statusCode indicates an error situation. For sub-group not created by the group hosting SCL this shall be membersContentResponses which contains collection of responses responded from members of this sub-group.

11.5 Resource Attributes

The M2M system shall support the following Attributes.

The type and values shall be supported according to the description given in table 11.36.

The default column specifies the default value of the attribute as set by hosting SCL in the resource representation, if the CREATE or UPDATE request did not include a value for that attribute, or if the provided value was unacceptable, and the hosting SCL is allowed to choose a value.

Table 11.36: Resource attributes

Name	Type	Default	Description
accessRightID	AnyURI	No default. If absent, all the entities that correspond to ancestor resources shall have the full set of permissions.	ID of an access rights resource Example: "http://*platform1.service_provider1.m2m_service.network_operator/accesRrights/right1*".
aPoC	AnyURI	No default. If absent, scl retargeting shall not allowed for the application.	The Application Point of Contact is a URI that identifies how requests are retargeted. Retargeting is described in clause 10.3.2.23.
aPocHandling	APocHandling	SHALLOW.	This attribute determines the way how the scl retargets request to the application based on the aPoc attribute and the aPoCPaths attribute. See clause 10.3.2.23.
aPoCPaths	APocPaths	No default.	The ApplicationPocPaths, is used to determine if a request to targetID is to be retargeted, taking into account the aPocHandling attribute, see [SCL retargeting to an application] for details.

Name	Type	Default	Description
appId	AnyURI	No default. The hosting SCL shall assign a globally unique ID in case no ID is provided in the CREATE request.	The ID of the application.

announceTo	AnnounceTo	Default is determined by server policies.
		If not present in the request, the SCL shall decide to which SCLs the resource will be announced. In the initial CREATE response the empty scl list shall be returned in the ResponseConfirm. The activated element shall be set to TRUE. The global attribute shall be absent, and by default it shall be FALSE.
		<announceTo>
		  <activated>TRUE</activated>
		  <sclList/>
		</announceTo>
		In an UPDATE request, the same applies, but in case the resource is already announced to some SCLs, the sclList shall indicate the list of SCLs already announced-to.
		<announceTo>
		  <activated>TRUE</activated>
		  <sclList>
		<scl>http://scl1.m2mprovider.org/</scl>
		<scl>http://scl2.m2mprovider.org/</scl>
		  </sclList>
		</announceTo>
		After the hosting SCL has performed some or all of the announcements, it shall update the attribute.
			In a request on mIa or dIa, this is interpreted as the list of the SCLs that the SCL will try to announce to on behalf of the requestor. In responses, the list indicates the actual list of resources to which the resource is announced at the moment.
			If this attribute is not provided in requests on the mIa or dIa, the local SCL will decide where the resource will be announced.
			Example:
			<announceTo>
			  <activated>FALSE</activated>
			  <sclList>
			   <scl>http://scl1.m2mprovider.org/</scl>
			     <scl>http://scl2.m2mprovider.org/</scl>
			  </sclList>
			</announceTo>

channelData	ChannelData	No default.	Any data returned for a specific notificationChannel. This type of the channelData depends on the channelType. For the LONG_POLLING channelType, the data returned includes a longPolling URI.
channelType	ChannelType	No default. This attribute is mandatory in the CREATE request and UPDATE requests are not supported.	The type of the notificationChannel. Example: LONG_POLLING
cmdType	CmdType	No default. Each <mgmtCmd> has a specific cmdType	The command type of a <mgmtCmd> resource represent Example: "REBOOT"
consistencyStrategy	ConsistencyStrategy	ABANDON_MEMBER	Attribute belongs to a group resource. Indicates the operation needs to take if the memberType validation fails.
contact	AnyURI	No default. This attribute is mandatory in CREATE and UPDATE requests.	The URI where the subscriber wants to receive its notifications.

Name	Type	Default	Description
contactInfo	ContactInfo	No default, This attribute is mandatory in CREATE and UPDATE requests	The contact Information for the reaching the device. Example: <contactInfo><contactURI>http://myhost.com:8080 </contactURI</contactInfo>
contactURI	AnyURI	No default.	The hosting URI shall create a unique URI and return this in the response to the CREATE request for the notificationChannel.
content	Content	No default. This attribute is mandatory in CREATE requests.	Real content of an instance.
contentInstances FilterCriteria	ContentInstanc esFilterCriteria	No default.	This are criteria extended for contentInstances resources and sub-resources that filter the results. They shall either be used in a GET (as query parameters) or in a subscribe. Example: *<contentInstancesFilterCriteria>* *<fromSize>32</fromSize>* *<contentTypes>* *<contentType>DS18S20</contentType>* *< contentType>Celsius</ contentType>* *</contentTypes>* *</contentInstancesFilterCriteria>*
contentSize	Long	Generated by the hosting SCL. Set to the actual contentSize of the received content.	Size in bytes of a content instance
contentTypes	ContentTypes	No default. If the content is represented as a base64Binary, then the hosting SCL shall set the value to the value of the xmlmime:contentType attribute. If the content is represented as raw content, then the hosting SCL shall set the value to the contentType as received by the transport layer. In case the top-level contentType is multipart/alternative, then the hosting SCL shall include all the alternative ontentType in that multipart, including the multipart/alternative itself.	Content-Type of the content instance. Example: "image/png" Content-Types of the *contentInstance* resource. Example: <contentTypes> <contentType>multipart/alternative</contentType> <contentType>image/jpeg</contentType> <contentType>image/png</contentType>< contentTypes>
creationTime	DateTime	Generated by the hosting SCL. Set to the actual time of creation of the containing resource.	Time of creation of a resource Example "*2002-10-10T17:00:00.000+05:00*"
currentByteSize	Long	Generated by the hosting SCL. Set to the actual number of bytes occupied by the instances in the containing resource.	Current size in bytes of data stored in a container resource. It is limited by the maxByteSize
currentNrOfInsta nces	Long	Generated by the hosting SCL. Set to the actual number of instances in the containing resource.	Current number of instances in a container resource. It is limited by the maxNrOfInstances
currentNrOfMem bers	Long	Generated by the hosting SCL. Set to the actual number of memberIDs in the members attribute of the containing group resource.	Current number of members in a group. It is limited by the maxNrOfMembers

Name	Type	Default	Description
delayTolerance	DateTime	No default. If absent, the hosting SCL may decide when to schedule the notification, based on server policies, like load and subscriptions with a higher priority.	The time before the addition of the containing <instance> resource shall be notified to any subscribers. Example "*2002-10-10T17:00:00.000+05:00*"
discoveryURI	AnyURIList	Empty list.	A list of discovered URIs. It may be an empty list.
description	String	No default.	This attribute relates to <mgmtObj> resources and is a text format description of mgmtObj
execDisable	AnyURI	Empty.	Used to trigger to cancel an <execInstance>
execEnable	AnyURI	Empty.	Used to trigger to execute a <mgmtCmd>
execReqArgs	ExecReqArgsList	No default.	The list of request arguments, specific to a <mgmtCmd> resource
execResult	ExecResultList	No default.	List of execution results of an <execInstance> It may be empty or have values for: startTime and completeTime, respectively
execStatus	ExecStatus	INITIATED	The execution status of an <execInstance> resource Example: "STARTED"
expirationTime	DateTime	Default determined by server policy. If a value is provided, the server shall try to find an acceptable value that is as close as possible to the requested value.	Expiration time for a resource Example "*2002-10-10T17:00:00.000+05:00*"
filterCriteria	FilterCriteria	No default.	This are criteria that filter the results. They shall either be used in a GET (as query parameters) or in a subscribe Example: <filterCriteria> <lastModifiedSince>*2002-10-10T17:00:00.000+05:00</lastModifiedSince>* *<searchStrings>* *<searchString>tag1</searchString>* *<searchString>some other tag</searchString>* *</searchStrings>* *</filterCriteria>*
href	AnyURI	Generated by the hosting SCL when the resource is created.	URI of the resource representation that contains this attribute.
id	Token	Generated by the hosting SCL at resource creation.	Identity of a resource to be created. Example: "*thermometer*"
integrityValResults	IValResult	No default.	Contains three elements including : The raw integrity validation results binary list of "Pass (1) or Fail(0)" for component to functionality mapping defined by M2M SP. Signed integrity validation results Secure time-stamp Example <integrityValResults > < ivalResults> 123456 </ ivalResults> <signedIvalResults>146346 </signedIvalResults> <timeStamp> *2002-10-10T17:00:00.000+05:00* </timeStamp> </integrityValResults >
lastModifiedTime	DateTime	Generated by the hosting SCL. Set to the time of the last modification.	Last modification time of a resource Example: Example "*2002-10-10T17:00:00.000+05:00*"

Name	Type	Default	Description
link	AnyURI	No default. This attribute is mandatory in CREATE request and ignored in the UPDATE requests	Link to the resource that is being announced.
locationContainerType	LocationContainerType	No default. This attribute is mandatory in CREATE request.	The source of the location information Example: "LOCATION_SERVER_BASED"
locRequestor	String	No default. Location requestor identity is not available.	The identity of the <application> to be used for the content of privacy control when requesting the location information of a remote M2M Device or Gateway. The format of this attributed shall conform to the interface provided by the location server (e.g. MSISDN for a 3GPP location server). Example: "380561234567"
locTargetDevice	String	No default. Location target device identity is not available.	The device address to be used for retrieving the location information of the M2M Device or Gateway which is represented by this <scl>. This attribute is only used in the case that the location information is provided by a network-based location server (e.g. a 3GPP location server). It will be provided to the location server by the hosting SCL (i.e. NTOE) for the location information retrieval. The format of this attributed shall conform to the interface provided by the location server (e.g. MSISDN of a target device Example: "380561234567"
matchSize	Long	0	The number of resources matched the filterCriteria.
maxByteSize	Long	Default determined by server policy. If a value is provided, the server shall try to find an acceptable value that is as close as possible to the requested value.	Maximum number of bytes that is allocated for a container resource for the overall instances Predefined values: "*-1*": dynamic size
maxInstanceAge	Duration	Default determined by server policy. If a value is provided, the server shall try to find an acceptable value that is as close as possible to the requested value.	Maximum age of instances of a container resource, the value is expressed in seconds Predefined values: a negative duration indicates unlimited duration
maxNrOfInstances	Long	Default determined by server policy. If a value is provided, the server shall try to find an acceptable value that is as close as possible to the requested value.	Maximum number of instances of a container resource Predefined values: "*-1*": dynamic number of instances
maxNrOfMembers	Long	Default determined by server policy. If a value is provided, the server shall try to find an acceptable value that is as close as possible to the requested value.	Maximum number of members for a group resource Predefined values: "*-1*": dynamic number of members
members	AnyURIList	Empty list.	List of members in the group.
membersContentResponses	MembersContentResponses	Empty list.	
memberType	MemberType	No default. This attribute is mandatory in CREATE request and ignored in the UPDATE requests	Type of group member Example: APPLICATION
mgmtProtocolType	MgmtProtocolType	No default	It is defined and used to store the management protocol that this *<scl>* supports. Example: "OMA DM v1.2"

Name	Type	Default	Description
minimalTimeBetweenNotifications	Long	No default. If absent, the server can send a notification independent on when the previous notification was sent.	Minimal time between notifications in milliseconds.
moId	AnyURI	No default	Contains a URN that uniquely identifies the MO data model used for this *<mgmtObj>* resource as well as the manage function and version it represents. This attribute shall be provided during the creation of the *<mgmtObj>* and shall not be modifiable afterwards.
onlineStatus	OnlineStatus	No default. For <scl> resource: the value is computed as shown in the primitive descriptions For <m2mPoc> resources: the default is ONLINE	Status of SCL. Indicates if the SCL is reachable for M2M REST traffic. For the <scl> resource under the NSCL sclBase tree the status is set by the hosting SCL based on the provided m2mPoc information and/or long polling activity. If the <*scl*> resource contains at least one active (online) m2mPoc, then the onlineStatus of that SCL resource shall be set to ONLINE. If the <*scl*> resource is currently involved in long-polling, the online status of that SCL resource shall be set to be ONLINE. If there are no m2mPocs defined or if all m2mPocs are marked as OFFLINE and no long-polling is ongoing, then the onlineStatus of that SCL shall be set to OFFLINE. If there are m2mPocs, and all of them are marked as NOT_REACHABLE, the onlineStatus of the SCL shall be set to NOT_REACHABLE to indicate that the SCL cannot be reached using any of the m2mPocs. NOT_REACHABLE shall be regarded as a sub-state of ONLINE. For the corresponding D/G SCL, the onlineStatus is known by the D/G SCL itself, and it is set accordingly.
originalMO	AnyURI	No default.	Contains the local path of the original MO instance on the remote entity which is represented by the <mgmtObj> resource in the hosting SCL. This attribute shall be provided during the creation of the <mgmtObj>, so that the hosting SCL can correlate the created <mgmtObj> with the original MO instance on the remote entity for further REM operations. It shall not be modifiable after creation. The format of this attribute shall be a local MO path in the form as specified by existing management protocols (e.g. "./anyPath/Fw1" in OMA-DM, "Device.USBHosts.Host.3." in TR069).

Name	Type	Default	Description
permissions	Permissions	Defaults to the empty list of permissions.	List of enabled permissions Example: <permissions> <permission id="first"> <permissionFlags> <flag>READ</flag> <flag>WRITE</flag> </permissionFlags> <permissionHolders> <holderRefs><holderRef>http://m2m.operator.org/groups/group1</holderRef> <holderRef>http://gw1.operator.org/base</holderRef> </holderRefs> <all/> </permissionHolders> </permission> <permission id="second"> <permissionFlags> <flag>READ</flag> <flag>WRITE</flag> </permissionFlags> <permissionHolders> <holderRefs> <holderRef>http://m2m.operator.org/groups/group2</holderRef> </holderRefs> <domains> <domain>http://m2m.operator.org/scls/someScl</domain> <domain>http://another.operator.org/</domain> <domains> </permissionHolders> </permission> </permissions>
pocs	AnyURIList	Empty list.	Point of contact of an interface Example: "ftp://192.168.2.101/ftproot"
primitiveType	PrimitiveType	No default.	Type of primitive constant identifier (read only) for a correct primitive detection to ensure appropriate local processing of a received primitive Example: ACCESS_RIGHTS_CREATE_REQUEST
remTriggerAddr	AnyURI		Contains the "triggering address" of the remote entity represented by the M2M Device's or Gateway's <*scl*> registered resource with a hosting network <sclBase> for management purpose. Example: Example: "http://*platform1.service_provider1.m2m_service.network_operator/device1/* *or* "http://*platform1.service_provider1.m2m_service.network_operator/gateway1/*
representation	AnyType	No default.	The resource representation of the modified resource. In case of errors this will be the representation of the error information.
resourceURI	AnyURI	Generated by the hosting SCL after resource creation.	The URI of a resource Example: "*platform1.service_provider1.m2m_service.network_operator/resource*"

Name	Type	Default	Description
schedule	Schedule	No default. If absent, the server cannot determine when the device will be next available.	Schedule that tells when an SCL is available.
sclId	AnyURI	No default. This is a mandatory attribute in create and update requests	The globally unique ID of the SCL.
searchStrings	SearchStrings	Empty list.	Tokens used as keys for searching resources and contents Examples: "*meter*", "*valve*"
selfPermissions	Permissions	No default. This attribute is mandatory in CREATE and UPDATE requests	List of permissions that apply to the containing access right resources itself.
serverCapability	Boolean	No default.	The value of serverCapability is set to TRUE only if there are m2mPocs available. If TRUE the registered SCL is server capable, i.e. can handle incoming requests.
statusCode	StatusCode		Local processing statusCode Example: "*STATUS_CREATED*"
subscriberID	AnyURI	No default. This attribute is mandatory in CREATE and UPDATE requests	The id of the subscriber.
subscriptionReference	AnyURI	No default.	The reference to the subscription URI that corresponds to the notification.
subscriptionType	SubscriptionType	Generated by the hosting SCL depending on the type of contactURI. If the contact refers to a notificationChannel, the subscriptionType shall be set to SYNCHRONOUS. If the contact does not refer to a notificationChannel, then the subscriptionType shall be set to ASYNCHRONOUS.	Indicates whether the subscription is synchronous or asynchronous. Example: <subscriptionType>ASYNCHRONOUS</subscriptionType>
truncated	Boolean[0..1]	FALSE	Indicates if the results of the discovery is truncated.

Resource attributes not specified in TS 102 690 [2], but introduced to maintain references to child resources. See annex B.

Table 11.37: Resource attributes for sub-resources

Name	Type	Default	Description
accessRightCollection	NamedReferenceCollection	Empty collection	List of references to accessRight resource
accessRightsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to accessRights resource
applicationCollection	NamedReferenceCollection	Empty collection	List of references to application resource
applicationsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to applications resource
containerCollection	NamedReferenceCollection	Empty collection	List of references to container resource

Name	Type	Default	Description
containersReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to containers resource
contentInstanceCollection	ContentInstanceCollection	Empty collection	List of contentInstance resource representations
discoveryReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to discovery resource
groupCollection	NamedReferenceCollection	Empty collection	List of references to group resource
groupsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to groups resource
memberContentsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to memberContents resource
mgmtObjReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to mgmtObj resource
mgmtObjsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to mgmtObjs resource
mgmtCmdReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to mgmtCmd resource
notificationChannelCollection	NamedReferenceCollection	Empty collection	List of references to notificationChannel resource
notificationChannelsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to notificationChannels resource
sclCollection	NamedReferenceCollection	Empty collection	List of references to scl resource
sclsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to scls resource
subscriptionCollection	NamedReferenceCollection	Empty collection	List of references to subscription resource
subscriptionsReference	AnyURI	Reference to child resource created by the hosting SCL when the parent is created.	Reference to subscriptions resource

11.6 Primitive Attributes

The M2M system shall support the following primitive Attributes.

The type and values shall be supported according to the description given in table 11.38.

Table 11.38: Primitive attributes

Name	Type	Default	Description
requestingEntity	AnyURI		In case the issuer is an SCL, this shall be the URI of the <scl> resource. The <scl> resource which has the URI; <sclBase1>/scls/<SCLID>, where the <sclBase1> is the sclBase of the remote SCL where the SCL has registered or with which the SCL has registered. In case of an Application, this is the URI of the <application> I.e. <localSclBase>/applications/<applicationID> resource corresponding to the application registration
errorInfo	ErrorInfo	No default value	This provides additional information about the error that occurred. EXAMPLE: <errorrInformation> <statusCode>STATUS_BAD_REQUEST</statusCode> <additionalInformation>The mandatory parameter id is missing from the CREATE request</additionalInformation> </errorInformation>
targetID	AnyURI	No default value	ID of the resource targeted by primitives requesting access to a resource.
primitiveType	PrimitiveType	No default value	Attribute indicating the type of the primitive.
statusCode	StatusCode	No default value	Attribute that indicates the status in primitives in response to requests.
searchPrefix	AnyURI	No default value	The path under which the discovery procedure needs to be executed.
maxSize	Long	No default value	Specifies the maximum number of discovered resources that can be returned during a discovery procedure
filterCriteria	FilterCriteria	No default value	Attribute to primitives for discovery procedures
contentInstancesFilterCriteria	ContentInstancesFilterCriteria	No default value	Specialization of filterCriteria specifically used for retrieving a contentInstances resource
TRPDT	TrpdtType	No default value	Attribute to RequestIndication primitives for access to remotely hosted resources.
RCAT	RcatType	No default value	Attribute to RequestIndication primitives for access to remotely hosted resources.

12 Security Message Definitions

12.1 PANA AVPs

Following vendor-specific PANA AVPs are defined in the present document. These AVPs shall include the Vendor-Id field in the AVP header and set its value to 13019 (IANA-assigned Private Enterprise Number for ETSI).

12.1.1 M2M-Usage-Type AVP

Table 12.1: M2M-Usage-Type AVP

Description	This AVP indicates the context of the PANA message
Value type	Enumerated
Value	One-octet enumeration with the following values: 1 = "M2M Bootstrapping" 2 = "M2M Erase" 3 = "M2M Connection Setup" 4 = "M2M Connection Tear-down" 0, and 5 to 255 = Reserved for future use

12.1.2 M2M-Bootstrap-Result AVP

Table 12.2: M2M- Bootstrap-Result AVP

Description	This AVP indicates the result of bootstrap procedure
Value type	Enumerated
Value	One-octet enumeration with the following values: 0 = "Success" 1 = "Failure" 2 to 255 = Reserved for future use.
NOTE: Value 0 ("Success") shall not be used if the PANA Result-Code AVP indicates failure (i.e. carries a value other than 0).	

12.1.3 M2M-Connection-Result AVP

Table 12.3: M2M-Connection-Result AVP

Description	This AVP indicates the result of connection procedure
Value type	Enumerated
Value	One-octet enumeration with the following values: 0 = "Success" 1 = "Failure" 2 to 255 = Reserved for future use
NOTE: Value 0 ("Success") shall not be used if the PANA Result-Code AVP indicates failure (i.e. carries a value other than 0).	

12.1.4 M2M-Node-ID AVP

Table 12.4: M2M-Node-ID AVP

Description	This AVP carries an M2M Node Identifier value
Value type	Binary
Value	16-octet value

12.1.5 M2M-MSBF-ID AVP

Table 12.5: M2M-MSBF-ID AVP

Description	This AVP carries an MSBF Identifier value.
Value type	Variable
Value	Type-specific value according to the following one-octet Types: 0 = "FQDN". Type field followed by the FQDN value (a NULL-terminating String) 1 = "IPv4 address". Type field followed by IPv4 address value (Unsigned32) 2 = "IPv6 address". Type field followed by IPv6 address value (Unsigned128) 3 to 255 = Reserved for future use

12.1.6 M2M-NSCL-ID AVP

Table 12.6: M2M-NSCL-ID AVP

Description	This AVP carries an NSCL Identifier value
Value type	String
Value	NULL-terminated string e.g. the sclBase URI of the NSCL

12.1.7 M2M-SP-ID AVP

Table 12.7: M2M-SP-ID AVP

Description	This AVP carries an M2M Service Provider Identifier value
Value type	String
Value	An FQDN in the form of a NULL-terminated String

12.1.8 M2M-Connection-ID AVP

Table 12.8: M2M Connection-ID AVP

Description	This AVP carries an M2M Connection Identifier value
Value type	Binary
Value	4-octet value

12.1.9 M2M-Encr-Encap AVP

Table 12.9: M2M Encr-Encap AVP

Description	This AVP carries one or more AVPs in encrypted form
Value type	Composite
Value	Two-octet Nonce counter value. Starts from 1 and monotonically incremented each time this AVP is sent/received Variable-length octets carrying the encrypted AVPs using the AES-CTR algorithm with the Nonce and the K_{PE} values

12.1.10 M2M-IBE-Params AVP

Table 12.10: M2M-IBE-Params AVP

Description	This AVP carries IBE parameters needed by the Device/Gateway for EAP-IBAKE.
Value type	Composite
Value	Two-octet IBEEC_p_size carrying the size of IBEEC_p in bits. Variable-octet Binary IBEEC_p carrying the large prime number that describes the IBE elliptic curve of table 6.1. Length of this field is indicated by the IBEEC_p_size value. Variable-octet Binary IBEEC_q carrying the prime number that divides (IBEEC_p+1). Length of this field is indicated by the IBEEC_p_size value. Variable-octet Binary P_kgf carrying the chosen point on IBEEC of table 6.1 by KGF. The length of this field is indicated by the IBEEC_p_size value. Variable-octet Binary P_kgf_pubId carrying the point on IBEEC of table 6.1 corresponding to IBE public key of KGF. The length of this field is indicated by the IBEEC_p_size value. Variable-octet Binary IBE_privKey carrying the private key corresponding to K_PUBp (see table 6.2). The length of this field is indicated by the IBEEC_p_size value. Variable-octet Binary P_NIST carrying the point P on the NIST curve of table 6.1. The length of this field is indicated by the NIST curve definition of the table 6.1. NULL-terminating String UTCDate carrying the date information used in K_PUB derivation (see table 6.2) in the form of YYYYMMDDHHMMSS.

12.1.11 M2M-DSCL-ID AVP

Table 12.11: M2M- DSCL-ID AVP

Description	This AVP carries an D/GSCL Identifier value
Value type	String
Value	NULL-terminated string e.g. the sclBase URI of the D/GSCL

12.1.12 M2M-MID-SEC AVP

Table 12.12: M2M- MID-SEC AVP

Description	This AVP carries mId security methods supported (when sent by the network) and selected (when sent by the D/G M2M Node)
Value type	Composite
Value	2-octet bit-flag. Following bit values are defined: Bit 0: Set to 1 if access network based security is supported, set to 0 otherwise. Kmc keys are not used if this flag is set to 1. Bit 1: Set to 1 if channel security is supported, set to 0 otherwise. Bit 2: Set to 1 if Kmc is supported for channel security, set to 0 otherwise. Bit 3: This bit is reserved for future use. If the Bit 1 is set to 0, then the Bit 2 shall be set to 0 by the sender and ignored by the receiver. If the Bit 1 is set to 1 then the Bit 2 shall be set to 1. Bit 3 shall be set to 0 by the sender and ignored by the receiver. Bit 4: Set to 1 if object security is supported, set to 0 otherwise. Bit 5: Set to 1 if Kmc is supported for object security, set to 0 otherwise. If the Bit 4 is set to 0, then the Bit 5shall be set to 0 by the sender and ignored by the receiver. If the Bit 4 is set to 1, then the Bit 5 shall be set to 1 by the sender. Bits 6-15: These bits are reserved for future use. These bits shall be set to 0 by the sender and ignored by the receiver. One or more of Bits 0, 1 and 4 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node.

12.1.13 M2M-XML-ALGOS AVP

Table 12.13: M2M-XML-ALGOS AVP

Description	This AVP carries XML security algorithms supported (when sent by the network) and selected (when sent by the D/G M2M Node).
Value type	Composite.
Value	1-octet bit-flag for XML-ENC Block Encryption algorithms. Following bit values are defined: Bit 0: Set to 1 if AES-128-GCM is supported, set to 0 otherwise. Bit 1: Set to 1 if AES-256-GCM is supported, set to 0 otherwise. Bits 2 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-ENC Key Transport algorithms. Following bit values are defined: Bit 0: Set to 1 if RSA-v1.5 is supported, set to 0 otherwise. Bit 1: Set to 1 if RSA-OAEP is supported, set to 0 otherwise. Bits 2-7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-1 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-ENC Symmetric Key Wrap algorithms. Following bit values are defined: Bit 0: Set to 1 if TRIPLEDES is supported, set to 0 otherwise. Bit 1: Set to 1 if AES-128 is supported, set to 0 otherwise. Bit 2: Set to 1 if AES-256 is supported, set to 0 otherwise. Bit 3: Set to 1 if AES-192 is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this AVP is sent by the network. One or none

	of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-ENC Message Digest algorithms. Following bit values are defined: Bit 0: Set to 1 if SHA1 is supported, set to 0 otherwise. Bit 1: Set to 1 if SHA256 is supported, set to 0 otherwise. Bit 2: Set to 1 if SHA512 is supported, set to 0 otherwise. Bit 3: Set to 1 if RIPE-MD160 is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-ENC Canonicalization algorithms. Following bit values are defined: Bit 0: Set to 1 if Canonical XML is supported, set to 0 otherwise. Bit 1: Set to 1 if Canonical XML with Comments is supported, set to 0 otherwise. Bit 2: Set to 1 if Exclusive XML Canonicalization is supported, set to 0 otherwise. Bit 3: Set to 1 if Exclusive XML Canonicalization with Comments is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-SIG Digest algorithms. Following bit values are defined: Bit 0: Set to 1 if SHA1 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG Encoding algorithms. Following bit values are defined: Bit 0: Set to 1 if base64 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG MAC algorithms. Following bit values are defined: Bit 0: Set to 1 if HMAC-SHA1 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG Signature algorithms. Following bit values are defined: Bit 0: Set to 1 if DSAwithSHA1 is supported, set to 0 otherwise. Bit 1: Set to 1 if RSAwithSHA1 is supported, set to 0 otherwise. Bits 2-7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-1 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node. 1-octet bit-flag for XML-SIG Canonicalization algorithms. Following bit values are defined: Bit 0: Set to 1 if Canonical XML 1.0 is supported, set to 0 otherwise. Bit 1: Set to 1 if Canonical XML 1.0 with Comments is supported, set to 0 otherwise. Bit 2: Set to 1 if Canonical XML 1.1 is supported, set to 0 otherwise. Bit 3: Set to 1 if Canonical XML 1.1 with Comments is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node.

	1-octet bit-flag for XML-SIG Transform algorithms. Following bit values are defined: Bit 0: Set to 1 if XSLT is supported, set to 0 otherwise. Bit 1: Set to 1 if XPATH is supported, set to 0 otherwise. Bit 2: Set to 1 if Enveloped Signature is supported, set to 0 otherwise. Bits 3 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-2 shall be set to 1 when this AVP is sent by the network. One or none of those bits shall be set to 1 when sent by the D/G M2M Node.

12.1.14 M2M-Erase-Token AVP

Table 12.14: M2M-Erase-Token AVP

Description	This AVP carries parameters required for performing M2M Node Bootstrap-Erase Procedure.
Value type	Composite
Value	TBD-octet M2M Node ID carrying the identifier of the M2M Node being deleted. 4-octet Key index carrying the Kmr index. 4-octet Nonce. 1-octet Type. Following values are defined: Reserved Network-initiated erase request D/G-initiated erase request Erase successful Erase rejected due to local policy 5 to 255 Reserved for future use 32-octet Hash.

12.1.15 M2M-KMR-Index AVP

Table 12.14a: M2M-KMR-Index AVP

Description	This AVP carries the M2M Kmr index value
Value type	Binary
Value	8-octet value

12.2 Security Resource Attributes

The security resource attributes starting shall be supported for D/G M2M Nodes, MSBF and MAS supporting the M2M Service Bootstrapping Parameter Delivery Procedure (used in the GBA-Based M2M Service Bootstrapping Procedure and the M2M Service Bootstrapping Procedure using TLS over TCP) and/or the M2M Service Connection Parameter Delivery Procedure (used in the M2M Service Connection Procedure based on TLS-PSK).

The type and values shall be supported according to the description given in table 12.15.

The type definitions shall conform to clauses 11.2, 11.3 and 11.4.

NOTE: These attributes have no default value.

Table 12.15: Security resource attributes

Name	Type	Description
securityConnectionId	HexBinary	64-bit M2M-Connection-ID assigned by the MAS during M2M Service Connection Parameter Delivery Procedure.
securityKmcIndex	UnsignedInt	Kmc-Index assigned to the Kmc during M2M Service Connection Parameter Delivery procedure.
securityKmrIndex	UnsignedInt	Kmr-Index assigned to the Kmr during M2M Service Bootstrap Parameter Delivery procedure.
securityLifetime	DateTime	In the case of an M2M Service Bootstrap Parameter Delivery procedure this attribute shall be the lifetime of the M2M Root Key (Kmr) established in the corresponding M2M Service Bootstrap procedure. In the case of an M2M Service Connection Parameter Delivery procedure this attribute shall be the lifetime of the M2M Connection Key (Kmc) established in the corresponding M2M Service Connection procedure.
securityEncryptedM2MKey	HexBinary	In the case of an M2M Service Bootstrap Parameter Delivery procedure this attribute shall be the value of a 256-bit M2M Root Key (Kmr), encrypted using AES-256 Key Wrap algorithm under KmrWrapKey exported from the TLS master_secret. In the case of an M2M Service Connection Parameter Delivery procedure, this attribute shall be the value of a 256-bit M2M Connection Key (Kmc), encrypted using AES-256 Key Wrap algorithm under KmcWrapKey exported from the TLS master_secret.
securityMasFqdn	AnyUri	FQDN of an MAS.
securitymIdFlags	HexBinary	The securitymIdFlagsindicates mId Security methods. Used in the M2M Service Connection Parameter Delivery procedure. This attribute definition shall apply to both the securitymIdFlags primitive attribute (see clause 12.3) sent from D/G M2M Node to network) and securitymIdFlags security resource attribute sent from network from D/G M2M Node. The securitymIdFlag is 16-bits. Following bit values are defined: Bit 0: Set to 1 if access network based security is supported, set to 0 otherwise. Kmc keys are not used if this flag is set to 1. Bit 1: Set to 1 if channel security is supported, set to 0 otherwise. Bit 2: Set to 1 if Kmc is supported for channel security, set to 0 otherwise. Bit 3: This bit is reserved for future use. If the Bit 1 is set to 0, then the Bit 2 shall be set to 0 by the sender and ignored by the receiver. If the Bit 1 is set to 1, then the Bit 2 shall be set to 1 by the sender. The Bit 3 shall be set to 0 by the sender and ignored by the receiver. Bit 4: Set to 1 if object security is supported, set to 0 otherwise. Bit 5: Set to 1 if Kmc is supported for object security, set to 0 otherwise. If the Bit 4 is set to 0, then the Bit 5 shall be set to 0 by the sender and ignored by the receiver. If the Bit 4 is set to 1, then the Bit 5 shall be set to 1 by the sender. Bits 6-15: These bits are reserved for future use. These bits shall be set to 0 by the sender and ignored by the receiver. One or more of Bits 0, 1, and 4 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network.
sclIdList	AnyURIList	This attribute maps to a list of sclId resource attributes (clause 11.5). This attribute provides a list of NSCL-IDs (during M2M Service Bootstrap Parameter Delivery Procedure). This list of NSCL-IDs shall be used to determine the next point of contact.
securityXmlAlgorithmsFlags	HexBinary	This attribute carries XML security algorithms supported (when sent by the D/G M2M Node) and selected (when sent by the network). This attribute definition shall apply to both the securityXmlAlgorithmsFlags primitive attribute (see clause 12.3) sent from D/G M2M Node to network) and securityXmlAlgorithmsFlags resource attribute sent from network from D/G M2M Node. 1-octet bit-flag for XML-ENC Block Encryption algorithms. Following bit values are defined: Bit 0: Set to 1 if AES-128-GCM is supported, set to 0 otherwise. Bit 1: Set to 1 if AES-256-GCM is supported, set to 0 otherwise. Bits 2 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this attribute is sent by the D/G

Name	Type	Description
		M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-ENC Key Transport algorithms. Following bit values are defined: Bit 0: Set to 1 if RSA-v1.5 is supported, set to 0 otherwise. Bit 1: Set to 1 if RSA-OAEP is supported, set to 0 otherwise. Bits 2-7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-1 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-ENC Symmetric Key Wrap algorithms. Following bit values are defined: Bit 0: Set to 1 if TRIPLEDES is supported, set to 0 otherwise. Bit 1: Set to 1 if AES-128 is supported, set to 0 otherwise. Bit 2: Set to 1 if AES-256 is supported, set to 0 otherwise. Bit 3: Set to 1 if AES-192 is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-ENC Message Digest algorithms. Following bit values are defined: Bit 0: Set to 1 if SHA1 is supported, set to 0 otherwise. Bit 1: Set to 1 if SHA256 is supported, set to 0 otherwise. Bit 2: Set to 1 if SHA512 is supported, set to 0 otherwise. Bit 3: Set to 1 if RIPE-MD160 is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-ENC Canonicalization algorithms. Following bit values are defined: Bit 0: Set to 1 if Canonical XML is supported, set to 0 otherwise. Bit 1: Set to 1 if Canonical XML with Comments is supported, set to 0 otherwise. Bit 2: Set to 1 if Exclusive XML Canonicalization is supported, set to 0 otherwise. Bit 3: Set to 1 if Exclusive XML Canonicalization with Comments is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-SIG Digest algorithms. Following bit values are defined: Bit 0: Set to 1 if SHA1 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG Encoding algorithms. Following bit values are defined: Bit 0: Set to 1 if base64 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG MAC algorithms. Following bit values are defined: Bit 0: Set to 1 if HMAC-SHA1 is supported, set to 0 otherwise. Bits 1 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. Bit 0 shall be set to 1. 1-octet bit-flag for XML-SIG Signature algorithms. Following bit values are defined:

Name	Type	Description
		Bit 0: Set to 1 if DSAwithSHA1 is supported, set to 0 otherwise. Bit 1: Set to 1 if RSAwithSHA1 is supported, set to 0 otherwise. Bits 2-7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-1 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-SIG Canonicalization algorithms. Following bit values are defined: Bit 0: Set to 1 if Canonical XML 1.0 is supported, set to 0 otherwise. Bit 1: Set to 1 if Canonical XML 1.0 with Comments is supported, set to 0 otherwise. Bit 2: Set to 1 if Canonical XML 1.1 is supported, set to 0 otherwise. Bit 3: Set to 1 if Canonical XML 1.1 with Comments is supported, set to 0 otherwise. Bits 4 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0-3 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network. 1-octet bit-flag for XML-SIG Transform algorithms. Following bit values are defined: Bit 0: Set to 1 if XSLT is supported, set to 0 otherwise. Bit 1: Set to 1 if XPATH is supported, set to 0 otherwise. Bit 2: Set to 1 if Enveloped Signature is supported, set to 0 otherwise. Bits 3 to 7: Reserved for future use. Shall be set to 0 by the sender, and shall be ignored by the receiver. One or more of Bits 0 to 2 shall be set to 1 when this attribute is sent by the D/G M2M Node. One or none of those bits shall be set to 1 when sent by the network.

12.3 Security Primitive Attributes

The security primitive attributes shall be supported for D/G M2M Nodes, MSBF and MAS supporting the M2M Service Bootstrapping Parameter Delivery Procedure in clause 6.4 (used in the GBA-Based M2M Service Bootstrapping Procedure in clause 6.2.1 and the M2M Service Bootstrapping Procedure using TLS over TCP in clause 6.3.2) and/or the M2M Service Connection Parameter Delivery Procedure in clause 7.4.5 (used in the M2M Service Connection Procedure based on TLS-PSK clause 7.4).

The type and values shall be supported according to the description given in table 12.16.

The type definitions shall conform to clauses 11.2, 11.3 and 11.4.

NOTE: These attributes have no default value.

Table 12.16: Security primitive attributes

Name	Type	Default	Description
securityM2MNodeId	HexBinary	No default value	A128-bit M2M-Node-ID associated with the D/G M2M Node. This attribute is used in the M2M Service Bootstrap Parameter Delivery Procedure for procedures using HTTP (clause 6.4) incorporated into the GBA-Based M2M Service Bootstrap Procedure (clause 6.2.1) and the M2M Service Bootstrap Procedure using TLS/TCP (clause 6.3.2).
securitymIdFlags	HexBinary	No default value	This attribute indicates mId Security methods supported by the D/G M2M Node. This attribute is used in the M2M Service Connection Parameter Delivery procedure (clause 7.4.5) used in the M2M Service Connection Procedure based on TLS-PSK (clause 7.4). See definition of securitymIdFlags security resource attribute in clause 12.2.
sclId	AnyURI	No default value	This attribute is identical to the sclId resource attribute (clause 11.5). This attribute is used in various ways in the M2M Service Bootstrap Parameter Delivery Procedure for procedures using HTTP (clause 6.4) incorporated into the GBA-Based M2M Service Bootstrap Procedure (clause 6.2.1) and the M2M Service Bootstrap Procedure using TLS/TCP (clause 6.3.2) and M2M Service Connection Parameter Delivery procedure (clause 7.4.5) used in the M2M Service Connection Procedure based on TLS-PSK (clause 7.4).
securityXmlAlgorithmsFlags	HexBinary		This attribute indicates XML security options supported by the D/G M2M Node. This attribute is used in the M2M Service Connection Parameter Delivery procedure for TLS-Based procedures (clause 7.4.5) used in the M2M Service Connection Procedure based on TLS-PSK (clause 7.4). See definition of securityXmlAlgorithmsFlagssecurity resource attribute in clause 12.2.

Annex A (normative): General mapping of primitives

A.1 Introduction

This clause describes a general mapping of communication primitives described in clause 10 into specific protocol messages and vice versa, ready to be transferred over a data transfer media, e.g. a TCP/IP or UDP/IP protocol stack.

Specific primitives are specified in clause 10 with attributes specified in table 11.38.

For the present release the recommended binding for dIa, mId and mIa is HTTP; other bindings, such as CoAP can apply.

A.2 Architectural model

Figure A.1 shows the architectural model that is adopted to support the description of primitive mapping.

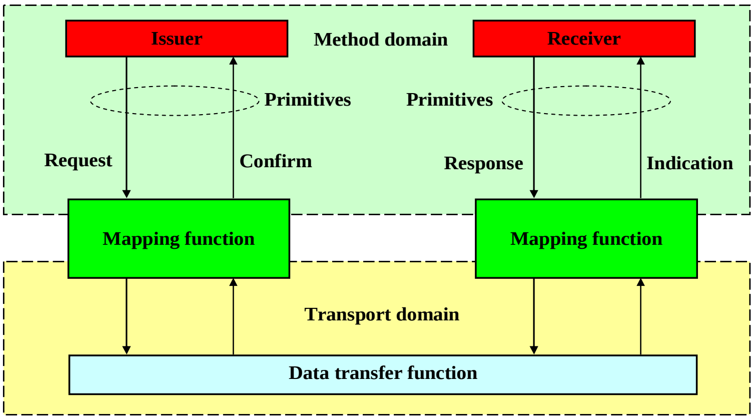


Figure A.1: Architectural model for primitive mapping

The model of figure A.1 implies the following assumptions:

- a method shall consist of one request/indication and one or two response/confirm primitives;
- if there are two response/confirm primitives in a method, the first shall be a response/confirm with statusCode STATUS_ACCEPTED;
- communication between an issuer and a receiver in the method domain shall be performed with the following primitives: request, indication, response and confirm;
- communication between an issuer and a receiver in the transport domain shall be performed using a data transfer function;
- mapping of primitives shall be performed by a mapping function that links both method and transport domains;
- primitives (request, response) shall be mapped to specific protocol messages for transmission into media domain;
- received specific protocol messages shall be mapped into primitives (confirm, indication) to be delivered in the method domain.

A.3 Primitives modelling

Primitives shall be are modelled as follows:

- a primitive shall be a data structure that describes with appropriate attributes a specific procedure request or answer in both issuer and receiver entities;
- a primitive shall consist of:
  - control part: description of attributes required for a procedure request or response;
  - an optional payload part: description of user data;
- a request primitive shall contain the attribute "<targetID>" in order to provide the destination where to deliver the mapped specific protocol message;
- the request and the indication shall be modelled as one primitive called "requestIndication" which is used in two directions (incoming and outgoing);
- the response and the confirm shall be modelled as one primitive called "responseConfirm" which is used in two directions (incoming and outgoing).

Figure A.2 shows the primitives modelling.

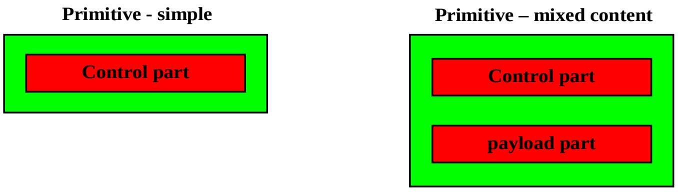


Figure A.2: Primitives modelling

In addition to performing just the mapping to the underlying protocol, the mapping functions in the transport domain may also handle some protocol specific parts of the interaction.

For example:

- caching of data;
- optimization of the interaction by using conditional requests;
- transparent handling of the semi-asynchronous and asynchronous communication models.

A.4 Mapping directives for request primitives

Mapping to a specific protocol message of a request primitive shall follow these rules:

- request primitive shall be provided to mapping function by the issuer entity;
- request primitive shall be correctly indicated;
- mapping of control data and payload into the appropriate format of the specific protocol shall be supported in the transport domain;
- mapped message shall be delivered by the data transfer function.

A.5 Mapping directives for indication primitives

Mapping from a received specific protocol message to an indication primitive shall follow these rules:

- received specific protocol message shall be provided to the mapping function by the receiver entity;
- specific protocol message content shall be identified;
- control data and payload shall be mapped to an indication primitive with addition of optional information attributes if required;
- indication primitive shall be provided to the receiver entity.

A.6 Mapping directives for response primitives

Mapping to a specific protocol message of a response primitive shall follow these rules:

- response primitive shall be provided to mapping function by the receiver entity;
- the correct operation to perform on the transport domain shall be identified on the response primitive;
- control data and payload shall be mapped into the appropriate format of the specific protocol supported in the transport domain;
- mapped message shall be delivered by the data transfer function.

A.7 Mapping directives for confirm primitives

Mapping from a received specific protocol message to a confirm primitive shall follow these rules:

- received specific protocol message shall be provided to the mapping function by the issuer entity;
- specific protocol message content shall be identified;
- control data and payload shall be mapped to a confirm primitive with addition of optional information attributes if required;
- confirm primitive shall be provided to the issuer entity.

Annex B (normative): Method and Ressources XSD formal definition

The XML Schema Description (XSD) files are contained in archive ts_102921v010201p0.zip which accompanies the present document.

B.1 Resource mapping to XML

B.1.1 Principles

The following principles shall be applied when mapping the datastructures in the present document to XML Scheme Definitions (XSDs) ([11] and [12]).

- Each resource shall be mapped to a single XSD file.
- All resources shall share the same namespace: http://uri.etsi.org/m2m
- Each enumerated type shall be mapped to an XSD enum.
- Each complex type shall be mapped to its own complex XML type.
- A resource representation shall be an XML document with the name of the resource type as its root element. Each resource type shall have a representation that contains all its attributes as a sequence of xml-elements In some specific cases a one of the attributes or elements may be mapped to an xml-attribute, specifically the id attribute shall be mapped to an xml-attribute called id.
- Each sub-element that is to be accessible using partial addressing will have its own root element definition that is just referenced in the containing structure. I.e. the representation contained in a partial retrieve, partial create of partial update has the XML document that contains the addressed attributename of element name as its root element.
- Each complex sub-element that appears in a collection shall have an id-xml-attribute to make it uniquely addressable using partial addressing.
- A collection resource shall contain an attribute which in turn shall contain all the references to the children. This allows the URLs of the children to be retrieved using partial addressing.
- A collection resource that can contain multiple types of child resources shall have such an attribute per type of child.
- Opaque content shall be mapped to base64 binary. Using XOP/MTOM ([63] and [64]) this can be mapped to mime attachments (RFC 2045 [14], RFC 2046 [15], RFC 2047 [16].

B.1.2 XSD files

In case of mismatch between the XSD files and the resources described in clause 10 the resource XSD shall lead.

1. accessRight.xsd
2. accessRightAnnc.xsd
3. accessRights.xsd
4. application.xsd
5. applicationAnnc.xsd
6. applications.xsd

7. attachedDevice.xsd
8. attachedDevices.xsd
9. bootstrapParamSet.xsd
10. common.xsd
11. commonDM.xsd
12. commonSecurity.xsd
13. connectionParamSet.xsd
14. container.xsd
15. containerAnnc.xsd
16. containers.xsd
17. contentInstance.xsd
18. contentInstances.xsd
19. discovery.xsd
20. errorInfo.xsd
21. etsiAnpPolicy.xsd
22. etsiAreaNwkDeviceInfo.xsd
23. etsiAreaNwkDeviceInfoInstance.xsd
24. etsiAreaNwkInfo.xsd
25. etsiAreaNwkInstance.xsd
26. etsiBattery.xsd
27. etsiBatteryInstance.xsd
28. etsiCapabilityAction.xsd
29. etsiCapabilityInstance.xsd
30. etsiDeviceCapability.xsd
31. etsiDeviceInfo.xsd
32. etsiFirmware.xsd
33. etsiFirmwareAction.xsd
34. etsiFirmwareInstance.xsd
35. etsiM2mSpPolicy.xsd
36. etsiMemory.xsd
37. etsiPerfLogAction.xsd
38. etsiPerformanceLog.xsd
39. etsiReboot.xsd
40. etsiRebootAction.xsd
41. etsiSafPolicySet.xsd

42. etsiSclMo.xsd
43. etsiSclMoAction.xsd
44. etsiSoftware.xsd
45. etsiSoftwareAction.xsd
46. etsiSoftwareInstance.xsd
47. etsiTrapEvent.xsd
48. etsiTrapEventAction.xsd
49. etsiTrapInstance.xsd
50. execInstance.xsd
51. execInstances.xsd
52. group.xsd
53. groupAnnc.xsd
54. groups.xsd
55. locationContainer.xsd
56. locationContainerAnnc.xsd
57. m2mPoc.xsd
58. m2mPocs.xsd
59. membersContent.xsd
60. mgmtCmd.xsd
61. mgmtObj.xsd
62. mgmtObjs.xsd
63. notificationChannel.xsd
64. notificationChannels.xsd
65. notify.xsd
66. parameters.xsd
67. responseNotify.xsd
68. scl.xsd
69. sclBase.xsd
70. scls.xsd
71. subscription.xsd
72. subscriptions.xsd
73. etsiRcatParamList.xsd

B.2 Data mapping to JSON

The mapping from XSD to JSON shall follow the rules defined in [9] in section 5.6.

B.3 Binary XML serialization

The M2M system supports the following XML serialization mechanism:

- Fast Infoset, see [10].
- Efficient XML (EXI), see [13].

B.4 Content type

The NSCL shall support the following content type:

- XML.
- JSON.
- Fast Infoset.
- EXI.

The M2M Device and M2M Gateway shall support at least one of the following content type:

- XML.
- JSON.
- Fast Infoset.
- EXI.

To ensure interoperability the M2M Device shall connect to a M2M Gateway which supports at least one of the content type supported by the M2M Device.

Annex C (normative): HTTP binding for M2M REST resources

This annex defines a mapping of the defined M2M REST resources to HTTP REST resources and XML data structures. The data types can be mapped to equivalent JSON data structures as well.

C.1 General

C.1.1 Resource Representation

Each normal HTTP REST resource shall have a representation in XML, see annex B or JSON (RFC 4627 [31]). One M2M resource corresponds to one XML or JSON datastructure.

For normal resources, the resource representation shall be carried in create requests (POST), update requests (PUT), successful retrieve (GET) responses and notify (POST) requests. It may also be carried in create (POST) and update (PUT) responses, but only if the hosting SCL decided to modify some of the resource attributes provided in the request. See the attribute handling rules and the tables in clause 9 for details.

When using partial addressing see clause C.3, a part/sub-tree of the resource representation shall be used instead of the full representation.

Virtual resources are resources that do not represent a specific resource representation as handled by the server, the virtual resources shall not have e-tags. Examples of virtual resources are discovery and membersContent.

For virtual resources, the data in the request and response shall not be resource representations. However for these resources a request and/or response datastructure is defined.

C.1.2 Content-type negotiation

Clients shall use the Accept Header in requests to indicate which resource representation is supported by its implementation.

The following content-types shall be used by the present document:

- *application/xml*: for XML encoded resource representations.
- *application/json:* for JSON encoded resource representations.
- *application/fastinfoset:* for binary encoded resource representations that are encoded using [10].
- *application/exi:* for binary encoded resource representations that are encoded using [13].
- *multipart/mixed:* for resource representations containing mime attachments (see below).

When transported in a HTTP request or HTTP response, the resource representation shall be carried in the body. Using XOP/MTOM encoding, some binary content (e.g. in the contentInstance representation and the membersContentResponses data structure) may be mapped to a multi-part mime structure (RFC 2046 [15]) if and only if mapped to json or xml. See [63] and [64] for details. In case the fastinfoset or exi representation is used, the binary content may not be encoded as mime, but embedded in the fastinfoset or exi representation as binary content.

C.1.3 Content-encoding

The NSCL shall support the use of Content-Encoding: gzip. This content-encoding shall not be used with any binary content-types, such as application/exi or application/fastinfoset.

C.1.4 Content-Location

A response to a CREATE method (POST) that contains a resource representation of the created resource shall include both a location header and a content-location header containing the same value; the URI of the created resource.

This tells the receiving party the representation in the response is really the representation of the created resource.

A response to an UPDATE method (PUT) that contains a resource representation shall include a content-location header of the updated resource, i.e. the same as the effective request URI.

This tells the receiving party the representation in the response is really the representation of the updated resource.

A response to a RETRIEVE method (PUT) that contains a resource representation that resides at a different location from the effective request URI, shall include a content-location header of the retrieved resource.

This specifically applies to the retrieval of a contentInstance via the 'latest' or 'oldest' URIs.

C.1.5 Conditional requests

A HTTP REST resource normally has a representation, which is what is returned as a result of a GET method. It is recommended that the server uses the e-tag mechanism defined in RFC 2616 [17] for these representations.

E-tags allow the use of conditional requests, using the if-match and if-none-match HTTP headers. If the server returns e-tags headers together with the resource representation, then the server shall support these conditional headers.

Regardless of e-tag support, the server shall support the last-modified time. This means that the server shall also support if-modified-since, if-unmodified-since condition headers in requests. Note that the lastModifiedTime is an attribute in most resource representations. In addition, the server shall also send the last-modified header in all responses that carry a resource representation that is not defined as a virtual resource. This header shall include the same date and time as the lastModifiedTime attribute in the corresponding resource.

It is recommended that a client that cached a previous version of the response uses conditional requests based on either the e-tag or the last-modified time, to both reduce the network load and to avoid concurrent modifications of a resource.

C.1.6 Caching

ETSI M2M may support caching by reusing the HTTP caching mechanism, as defined in [74].

The hosting SCL may generate the appropriate cache directive and caching related headers to enable caching in HTTP proxies with the following clarifications.

- Based on heuristics the hosting SCL can include an expirationTime header or max-age cache directive. The time should indicate how long the hosting SCL thinks the current representation will be valid. This time shall never exceed the expirationTime as stored in the resource representation.
- Due to acess rights the resource representation or response code depends on the requesting entity therefore, the cache-directive s-maxage or public should not be present in responses.
- If the hosting SCL cannot heuristicly determine a time for which the representation is valid, it may include an expiration time that is the current time or a max-age directive with value 0, in order to force revalidation by any proxy.
- Since the ContentInstance resource representation is immutable, it is recommended that the validity time of their representation heuristicly is derived from container parameters like maxInstanceAge and other heuristics controlling the lifetime of an instance (like average time between instance addition and maxNrOfInstances in the container). It shall never exceed the maxInstanceAge time.

Based on the caching related headers as described above, HTTP proxies can cache responses according to [74]

An intermediate SCL may go above and beyond the caching as described in [74] to provide a more M2M aware caching. It can use the caching as defined in [74] with the following clarifications and additions.

- The intermediate SCL may optionally implement the caching.

- Instead of explicitly validating a response by performing a GET request to the hosting SCL, the intermediate SCL may subscribe to the resource on the hosting SCL and keep its cached copies fresh in that way. In such a case the intermediate SCL/proxy does not have to (re-)validate with the hosting SCL.
  In such a case the responses would be stored in a shared cache, for additional clarifications on using shared cache, see also below.
  This also implies that the intermediate SCL is aware problems with the connection to the hosting SCL, i.e. it knows when its cache is no longer fresh.

  - It should be noted that when using a subscription mechanism for validation purposes, there may be a small delay between the resource on the hosting SCL being updated and the intermediate SCL that subscribed to the resource being informed about this. This time can be influenced by the delayTolerance in the subscription resource.

- The serving of stale resource by an intermediate is subject to the rules defined in [httpbis-6-caching], i.e. stale responses may only be served is the client indicated the willingness to accept them (controlled by the max-stale cache directive) and in case the response can not be validated.

- The intermediate SCL may, for performance reasons, store the caching information into a shared cache. Before serving a resource from such a shared cache, the intermediate SCL shall check if the requesting entity is allowed to retrieve the resource by checking the accessRight resource. The accessRight resource referenced by the accessRightID attribute should be cached as well, for optimal performance.

- If the URI of the resource contains query parameters, the responses are typically not cached unless explicitly indicated by the hosting SCL as defined in section 13.9 of [17]. However, if the intermediate SCL has obtained enough information to construct the response locally (e.g. if it can apply the filterCriteria on the cached non-filtered response), the intermediate SCL may respond with response that is derived from cached responses. Specifically, for contentInstances resources, the intermediate SCL may keep a fresh cache of all the contentInstance children as well as a cache of the contentInstances resource itself, in order create a resource representation of the contentInstances resource based on the local cached child resources, instead of going to the hosting SCL. This can be more efficient than caching responses per filterCriteria.

C.1.7 HTTP method mapping

The primitives shall be mapped as follows.

Table C.1: HTTP Method Mapping

Primitive type	HTTP METHOD
xxxRetrieveRequestIndication	GET
xxxUpdateRequestIndication	PUT
xxxCreateRequestIndication	POST
xxxDeleteRequestIndication	DELETE
xxxExecRequestIndication	POST (without a body)
xxxNotifyRequestIndication	POST (asynchronous notify) response to POST (long polling notify)
notificationChannelRequestIndication on the longPolling URI	POST without a body

However, for partial addressing, the mapping shall follow the mapping according to the method as defined in clause 10.39.

Table C.2: HTTP Method mapping for partial addressing

Method	HTTP METHOD
RETRIEVE	GET
UPDATE	PUT
CREATE	POST
DELETE	DELETE

Although HTTP/REST does allow CREATE to be implemented with a PUT, the present document shall not use this principle.

- Any PUT for a resource that does not exists shall be rejected with a "404 Not Found".
- Any PUT on an existing resource, with a "if-none-match: *" header shall be rejected with a "405 Method Not Allowed".

C.1.8 HTTP version

HTTP 1.1 shall be supported.

C.1.9 HTTP requestURI

The HTTP requestURI and host header shall be derived from the targetID primitive type attribute.

When addressing a remote resource on the mIa or dIa interface the targetID shall be used directly as the requestURI in HTTP. I.e. the local SCL shall act as an HTTP proxy for the request.

When addressing a local resource the host and port part of the targetID shall be used as the value for the host header. The abs_path shall be used as the requestURI.

For example, if the targetID has the following form:

- http://networkSCL.m2mprovider.org/applications/testApp.

Then on the mIa and dIa the following requestURI for a retrieve would look like this:

- GET http://networkSCL.m2mprovider.org/applications/testApp HTTP/1.1.

On the mId the requestURI would look as follows. Also a host header is added. So the first part of the request would look like this:

- GET /applications/testApp HTTP/1.1
  Host: networkSCL.m2mprovider.org

Any request URI parameters in the HTTP request URI shall be ignored, unless the HTTP mapping for the resource defines an explicit mapping.

C.2 Primitive mapping

C.2.1 Outgoing RequestIndication primitive to HTTP request

- The primitive type shall be mapped to an HTTP Method according to the table C.1..
- The HTTP request-line shall be constructed using the selected HTTP method, selected verb, the requestURI as described in clause C.1.9 and the HTTP version.
- For request addressing local resource or when using the mId, the host header shall be set according to the clause C.1.9.
- If the request primitive indicated a resource representation, then the content shall be included in the proper representation, including the correct Content-Type and Content-Length headers.
- The requestingEntity shall be included as a HTTP From header.

- If the request is a RETRIEVE request and contains *filterCriteria* or *contentInstancesFilterCriteria* as an primitive attribute, then each element in this filterCriteria shall be mapped as a query parameter in the requestURI, where the name of the query parameter shall be the element name of the *filterCriteria* or *contentInstancesFilterCriteria* element, and the value is an URI-encoded representation of the corresponding value as it would appear in an XML representation. The same query parameter can occur multiple times if the maxOccurs in the XML representation is more than one.
  For example, the filterCriteria

```
         <filterCritia>
              <searchString>test1</searchString>
              <searchString>test2</searchString>
         </filterCriteria>
    is mapped as
           ?searchString=test1&searchString=test2
```

- In case the RequestIndication primitive contains a TRPDT or RCAT attribute, the following mapping to HTTP headers shall be used to transport these attributes:
  - TRPDT shall be mapped to a header with a name field of "x-etsi-trpdt" using a string in the format of the TrpdtType as defined in clause 11.4 for the value field of the header.
  - RCAT shall be mapped to a header with a name field of "x-etis-rcat" using a string according to the definition of the enumerated RcatTyoe in clause 11.3 for the value filed of the header.
- If the request is a RETRIEVE and contains a primitive attribute maxSize or searchPrefix, then these shall be mapped as queryParameters with the same name in the requestURI. The value of the searchPrefix queryParameter is the URI encoded value of the searchPrefix. The maxSize value is the string representation of the long value.

C.2.2 Incoming HTTP response to responseConfirm primitive

The HTTP response is correlated with the outgoing request.

If the statusCode is in the range 2xx, then the response is considered a successful response. The body is then mapped to the corresponding resource representation or response data structure (for virtual resources).

The statusCodes shall be mapped as follows.

Table C.3: HTTP status Code mapping to primitive StatusCode

HTTP statusCode	StatusCode in request primitive
200, 203, 204, 205	"STATUS_OK"
201	"STATUS_CREATED"
202	"STATUS_ACCEPTED"
304	"STATUS_OK" Together with the locally cached resource representation

Any 2xx statusCode received that is not in the above list shall be treated as an error, and may be logged, but otherwise ignored. If the statusCode is 3xx, the response is logged, but otherwise ignored.

If the statusCode is 400 or higher, then this is an error and the body shall be interpreted as additional information about the error.

If the error response contains a body, the body shall be an errorInfo document, containing a statusCode element and an optional additionalInfo element. This is mapped to the errorInfo primitive attribute in the unsuccessful responseConfirm.

StatusCode 412, can be handled internally in the transport layer. See clause C.1.5.

The issuer and receiver are the reverse of the corresponding request primitive.

C.2.3 Incoming HTTP request to requestIndication primitive

If the requestURI includes an absolute URI and the host and port part of the URI do not match the host and port on which the request was received, then receiving SCL shall check if it has a secure association with the host and port indicated in the URI, and if so, extract the host and port part of the UI, map this as the host header and forward the request the request over that secure channel, with a requestURI matching only the path components of the original requestURI. If no secure channel is established with the indicated host and port, the request shall be rejected with TBD.

If the requestURI includes an absolute URI and the host and port part of the URI do match the host and port on which the request was received, the host header shall be ignored and the request shall be handled as shown below.

If the requestURI contains and absolute path, and the host header does not match the host and port on which the request was received, then the request shall be rejected with TBD.

If the requestURI contains and absolute path, and the host header matches the host and port on which the request was received, then the request shall be treated as shown below.

The primitive is derived from the combination of HTTP method and the type of the addressed resource as taken from the requestURI.

In general the following rules shall be applied.

Table C.4: Mapping HTTP request to primitives

HTTP Method	Body content	RequestURI	Primitive type
POST	Full <resourceType> representation	Parent resource for <resourceType>	<resourceType>createRequestIndication
GET	-	Resource of type <resourceType>	<resourceType>RetreiveRequestIndication
PUT	Full <resourceType> representation	Resource of type <resourceType>	<resourceType>UpdateRequestIndication
DELETE	-	Resource of type <resourceType>	<resourceType>DeleteRequestIndication
POST	Notify body reference to subscription child of <resourceType> and full <resourceType> representation	Any resource	<resourceType>NotifyRequestIndication
Partial addressing			
POST	Notify body reference to subscription child of <resourceType> and partial <resourceType> representation	Any resource	<resourceType>NotifyRequestIndication
POST	Partial <resourceType> representation	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication
GET	-	Resource of type <resourceType> with attribute accessor	<resourceType>RetreiveRequestIndication
DELETE	-	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication
PUT	Partial <resourceType> representation	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication

The receiver shall reject the request body that does not contain a resource representation that corresponds to the resource type identified by the requestURI.

Incoming OPTIONS requests are not listed in the table above, but shall be supported.

It is not mandatory to support incoming HEAD requests. It is not mandatory to support incoming TRACE requests.

The requestingEntity primitive attribute shall be included as a HTTP From header. If no From header is included in the request, the request is rejected with 402 statusCode.

The issuer and receiver shall be derived from the underlying secure communication channel.

C.2.4 Outgoing responseConfirm primitive to HTTP response

The primitive shall be correlated to the corresponding request primitive and the following rules shall be followed:

- The HTTP response is generated to the HTTP request corresponding to the request primitive.
- If the response primitive has any request data (resource representation or errorInfo), this shall be transported in the body of the response.
- The statusCode of the response shall be set according to table C.5.
- The resourceURI primitive attribute is transported in the location header of the HTTP response. This is applicable in case the statusCode primitive attribute is set to STATUS_CREATED.

For successful responseConfirm:

Table C.5: Primitive statusCode to HTTP statusCode mapping for successful codes

StatusCode responseConfirm primitive	StatusCode in HTTP
"STATUS_OK"	200 (if there is entity in the response) 204 (if there is no entity in the response)
"STATUS_CREATED"	201
"STATUS_ACCEPTED"	202 (see clause C.4)

For unsuccessful response confirm.

Table C.6: Primitive statusCode to HTTP statusCode mapping for unsuccessful codes

StatusCode in errorInfo	StatusCode in HTTP
"STATUS_ACCEPTED"	202 (see clause C.4)
"STATUS_BAD_REQUEST"	400
"STATUS_PERMISSION_DENIED"	401
"STATUS_FORBIDDEN"	403
"STATUS_NOT_FOUND"	404
"STATUS_METHOD_NOT_ALLOWED"	405
"STATUS_NOT_ACCEPTABLE"	406
"STATUS_REQUEST_TIMEOUT"	408
"STATUS_CONFLICT"	409
"STATUS_UNSUPPORTED_MEDIA_TYPE"	415
"STATUS_INTERNAL_SERVER_ERROR"	500
"STATUS_NOT_IMPLEMENTED"	501
"STATUS_BAD_GATEWAY"	502
"STATUS_SERVICE_UNAVAILABLE"	503
"STATUS_GATEWAY_TIMEOUT"	504
"STATUS_DELETED"	410
"STATUS_EXPIRED"	404

C.3 Partial addressing

In HTTP REST a resource is normally manipulated in its entirety. I.e. to provide a new representation, the old resource representation is replaced. Some modifications, e.g. additions are usually done with POST. However, HTTP REST does not define a generic mechanism for modifying the content within a resource in an idempotent way (POST is not idempotent, unless additional measures are taken). The HTTP PATCH method is defined in RFC 2616 [17], but the content of that message is unspecified. Furthermore, the HTTP PATCH method does not have any widespread support.

An alternative used in the present document is to allow the addressing of XML data inside the HTTP REST resource using principles similar to the spirit of REST, i.e. by allowing REST primitives to operate on embedded XML or JSON data. This is an approach popularized by XCAP (RFC 4825 [i.4]), which does allow arbitrary xpath expressions to manipulate parts of the representation of a resource. The approach taken here is very similar. I.e. the parts of the resource are identified using normal URIs, where the components correspond to the names of the attributes, or id-s of the attributes when selecting a sub-attribute in a collection.

In this way attributes can be removed (DELETE, if allowed), updated (PUT), created (PUT), values can be added to collections (POST). And of course the value of a single attribute can be obtained (GET).

Partial addressing is defined in clause 10.39.

The partial addressed attributes shall share the e-tags and last-modified-time of their containing HTTP REST resource. I.e. it can be seen as a filtering mechanism, to return or manipulate only a part of the data. However, this also means that conditional requests are based on the complete resource.

EXAMPLE: Suppose that resource {X} has two attributes called y and z.

Using partial addressing somebody can read only attribute y from resource {X}, i.e. using GET {X}/y.

This will return only the value of attribute y of resource {X} and an e-tag, say, 42.
Using partial addressing somebody now writes a new value for attribute z; PUT {X}/z with a body containing the new value of z. This returns a "204 No Content" response and an e-tag of 12.
The e-tag applies to the entire resource, so GET {X}/y with an if-none-match: 42 header will return the old value of y, even though this (filtered) part of the resource has not been changed.

C.4 Semi-asynchronous and asynchronous communication

In the interface towards the method domain (based on primitives) the communication model is mostly hidden. This section describes the specific actions that shall be taken by the transport layer in case semi-asynchronous or asynchronous communication is used.

C.4.1 Incoming RequestIndication Primitive and outgoing HTTP request

In case a requestIndication primitive is received and the entity (SCL or application) on which the transport resides is server capable the transport layer shall create an end-point identified by a unique URI in order to receive asynchronous callbacks and shall construct a HTTP request as in the normal case, however, now two additional HTTP headers shall be added to the request:

X-etsi-correlationID	set to a unique ID.
X-etsi-contactURI	set to the constructed end-point.

The transport shall maintain the relation between the end-point serving as the contactURI and the correlationID.

In case the entity (SCL or application) is not server capable, it shall only include the X-etsi-correlationID header and no x-etsi-contactURI shall be included.

The HTTP request constructed thus is send as described in the normal synchronous handling.

C.4.2 Incoming HTTP request and outgoing RequestIndication primitive

In case a HTTP request is received with an x-etsi-correlationID header, the outgoing primitive is constructed as described in clause C.2.3. However, additionally, the transport layer shall maintain the relation between the requestIndication primitive transaction and the combination of the correlationID, the requestingEntity and the contactURI (if present) received in the request.

Note that the correlationID is only unique with scope of the requestingEntity, so the tuple of collectionID and requestingEntity is used to generate a unique correlationID for internal use.

C.4.3 Incoming accept ResponseConfirm and outgoing HTTP response

In case the primitive user indicates that it is not yet interested or able to send a final response, but sends a responseConfirm primitive in the transaction with a statusCode STATUS_ACCEPTED, the transport layer shall send a HTTP response with a statusCode 202 Accepted.

On HTTP level, the request-response transaction shall be terminated.

However, the primitive user shall later send a final response to the same primitive request, so the *primitive* transaction is maintained until the final response.

C.4.4 Incoming HTTP accept response and outgoing ResponseConfirm

In case the transport layer receives a 202 accepted response to an ongoing HTTP request, it shall send a responseConfirm with statusCode STATUS_ACCEPTED to the primitive user that send the corresponding requestIndication.

If the outgoing HTTP request that corresponded to the requestIndication contained an x-etsi-contactURI, then the transport layer shall do nothing while waiting for the notify POST on that contactURI. However, when the transport layer can detect that the contactURI is no longer addressable, or reachable, it may switch to semi-asynchronous behaviour as described below and act as if the x-etsi-contactURI was not present in the original outgoing HTTP request. In this case the x-etsi-contactURI may be removed in the next re-send request.

If the outgoing HTTP request that corresponded to the requestIndication did not contain an x-etsi-contactURI, then the transport layer start a periodic polling timer. The timer value is determined by internal policies. On expiration of this timer the transport layer will send an outgoing polling request (see next clause). The transport layer shall maintain a relation between the timer and the original HTTP request and also to the original primitive requestIndication.

C.4.5 Outgoing HTTP polling request

In case the polling timer expires, the transport layer shall find the corresponding HTTP request, and retransmit that HTTP request to the receiver using exactly the same payload, and exactly the same x-etsi-correlationID header value as in the original request. The HTTP request shall not include an x-etsi-contactURI header. However, if the transport layer can detect that the issuer (Application or SCL) is now server capable, it may create an end-point for receiving notifications and include such an end-point in the x-etsi-contactURI header in the outgoing HTTP request. In this case it switches to asynchronous behaviour and shall stop the polling timer.

C.4.6 Incoming HTTP polling request

An incoming HTTP polling request is identified by the presence of an x-etsi-correlationID header value that corresponds to an ongoing primitive transaction or to a stored primitive result.

If there is no ongoing transaction or stored result the transport layer shall treat the request as a new request.

If there is an ongoing primitive transaction that has not yet resulted in a response, the transport layer shall respond with a 202 Accepted.

If the HTTP polling request includes an x-etsi-contactURI, this shall be stored by the transport layer and replaces any previously stored contactURI for this transaction.

If the primitive transaction that corresponds to the correlation id is completed and a response is stored, the transport layer shall send the stored response as a HTTP response to the polling request. Sending the result shall not delete the stored response. The reason behind this is that in case the response is lost and the request is retried, the response can still be returned. For non-idempotent operations this is essential!

The response shall be deleted a certain time after the primitive transaction completed, which is determined by internal server policy.

C.4.7 Incoming final ResponseConfirm

There are two cases in which a response shall be treated like described below:

1) In case the transport layer received the ResponseConfirm on a RequestIndication on which it already received an accepted response.

2) In case the transport layer received a ResponseConfirm to a RequestIndication primitive that resulted from a HTTP POST request (regardless of whether there was an accepted response before or not, i.e. this is also true in the synchronous case). The reason is that retransmits of non-idempotent requests shall result in the retransmission of the response to avoid executing the same request twice in the retransmission case.

In all other cases the ResponseConfirm shall be handled as specified in clause C.2.2.

For case 1, the handing shall depend on whether the last correlated HTTP request contained an x-etsi-contactURI or not.

In case the last HTTP request for this primitive transaction contained an x-etsi-contactURI, the transport layer shall create a notify request that encapsulates the response as shown below. It shall then send this as a POST to the URI indicated in the x-etsi-contactURI. In case this succeeds, i.e. in case there is a success response received to the notify, the transport layer shall not store the result.

In case the notify above fails, or in case the last HTTP request did not contain an x-etsi-contactURI, or in case 2 mentioned above, the transport layer shall store the received response for a duration that is dependant on internal policies.

Note that in case of a failure to deliver the notify, the transport layer may retry the notify, but the strategy it used for this is out of scope of the present document.

If the duration for which the response is stored expires, the response shall be deleted and the correlationID shall be forgotten. Any request that re-uses that correlationID shall be treated as a new request.

C.4.8 Mapping a ResponseConfirm to a notify POST

The response is mapped to a notify POST as follows.

The requestURI shall be set to the value in the x-etsi-contactURI header.

The body of the POST shall be an responseNotify entity, clause C.4.10 constructed as follows:

- The statusCode from the response is mapped to the statusCode attribute.
- The location header from the response is mapped to a locationHeader attribute.
- The e-tag header from the response is mapped to an etagHeader attribute.
- The body from the response is mapped to the representation attribute.
- The lastModified header is mapped to the lastModifiedHeader attribute.

The HTTP POST shall have a header called x-etsi-correlationID set the same value as the request that triggered the ResponseConfirm.

The resolving and routing shall be done as normal.

If a success HTTP response is received to a HTTP notify POST, the transport layer shall remove any stored response, remove all data related to the transactions and stop all the related timers.

If an error response is received to a notify POST, the transport layer may retransmit the notify POST according to internal policies.

C.4.9 Mapping a notify POST to a ResponseConfirm

The transport layer shall map a POST that is received on an end-point corresponding to a earlier set x-etsi-contactURI as follows.

It may check if the x-etsi-contactURI in the HTTP POST matches the one of the requests on which it has received an accepted response, but no final response yet. And it may send an error response on the POST with a 409 statusCode in case of a mismatch.

If there is a polling timer associated with the correlationID, this shall be stopped.

The responseNotify entity in the POST shall be mapped to a HTTP response:

- The statusCode element shall be mapped to the HTTP response code in the ResponseConfirm.
- The locationHeader element shall be mapped to the HTTP location header in the response (and ultimately to the resourceURI attribute in the ResponseConfirm primitive).
- The etagHeader element shall be mapped to the e-tag header in the response.
- The representation element shall be mapped to the body.
- The lastModifiedHeader element shall be mapped to the lastModified header in the response.

The resulting ResponseConfirm shall be treated in a similar way to a synchronously received response, i.e. it may lead to caching, etc.).

C.4.10 ResponseNotify entity

The responseNotify entity shall have the following structure.

Table C.7: ResponseNotify representation

AttributeName	Presence in subscriptionNotifyRequest	Description
statusCode	M	The statusCode from the corresponding response
representation	O	The body in the corresponding response, if there is a body in the response
locationHeader	O	The value of the location header in the response. This is present in case the statusCode indicates STATUS_CREATED
etagHeader	O	The value of the etag header in the response
lastModifiedHeader	O	The value of the lastModifiedHeader in the response

Table C.8: Resource Attributes for HTTP Mapping

Name	Type	Default	Description
locationHeader	String	No default	The value of the location header in the response
etagHeader	String	No default	The value of the etag header in the response
lastModifiedHeader	String	No default	The value of the lastModifiedHeader in the response

Annex D (normative): CoAP Binding for M2M REST Resources

This annex defines a mapping of the defined M2M primitives into Constrained Application Protocol (CoAP) [45] messages and XML data structures.

As CoAP supports a RESTful interaction model and a subset of HTTP functionality, this mapping is very similar to the HTTP mapping described in annex C.

D.1 Resource representation

This clause defines mappings of the defined M2M REST resources to CoAP REST resources and also provides CoAP resource representations encoded in XML. CoAP XML resource representations can also be mapped to equivalent JSON and EXI Content-Types which are also supported by CoAP.

Each normal CoAP REST resource shall have a representation in XML (annex B) or JSON (RFC 4627 [31]) or EXI [13]. One M2M resource shall correspond to one XML, JSON or EXI data structure.

For normal resources, the resource representation shall be carried in create requests (POST), update requests (PUT), successful retrieve (GET) responses and notify (POST or successful response to GET using CoAP Observe mechanism) requests. It may also be carried in create (POST) and update (PUT) responses, but only if the hosting SCL decided to modify some of the resource attributes provided in the request. When using partial addressing (clause 10.39), a part/sub-tree of the resource representation shall be used instead of the full representation.

Virtual resources are resources that do not represent a specific resource representation as handled by the server. The virtual resources do not return an Etag option. Examples of virtual resources are discovery and membersContent.

For virtual resources, the data in the request and response are not resource representations. However for these resources a request and/or response data structure is defined.

D.1.1 Content-type negotiation

The CoAP Accept header option shall be used for content-type negotiation. CoAP supports an Elective option called "Accept". When included one or more times in a request. Accept option can carry one or more media types, each of which is an acceptable media type for the client, in the order of preference. If no Accept option is given, the client does not express a preference. The client prefers the representation returned by the server to be in one of the media types, but is willing to accept the response also if the server returns a representation in a different media type.

The following CoAP Content-Types shall be used by the present document:

- ***.xml** *application/xml* for XML encoded resource representations.
- ***.json** *application/json* for JSON encoded resource representations.
- ***.exi** *application/exi* for binary encoded resource representations that are encoded using [13].

When transported in a CoAP request or CoAP response, the resource representation shall be carried in the CoAP payload. Each CoAP message only supports one Content-Type.

D.1.2 Conditional requests

A CoAP REST resource normally has a representation which is returned as a result of a GET method. A server may use the CoAP ETag option when returning a resource representation in a response. It is also recommended that a client that has one or more resource representations that it previously obtained from a server use the corresponding ETags in any subsequent requests to the same resource. Upon receiving a GET request that includes one or more ETag Options, the server inspects the ETags to determine whether any of the corresponding representations of the resource are current or not. If one is current, the server shall respond back with the ETag that is current along with a code of 2.03 - Valid. If none is current, the server shall respond with an updated representation of the resource and a response code of 2.05 - Content.

D.1.3 Caching

A CoAP client may choose to cache responses to reduce the response time and network bandwidth consumption for future equivalent requests to the same resource. The goal of caching in CoAP is for a client to reuse a prior response message to satisfy a current request. Based on the 'freshness' of the stored response, a client may reuse a cached response to service a new request. In doing so, sending a new request to the originating server can be avoided.

The mechanism for determining freshness is the CoAP Max-Age option. This option can be used by a server to specify an explicit expiration time for the response's resource representation. The Max-Age Option indicates that the response is to be considered not fresh after its age is greater than the specified number of seconds.

When a client has one or more cached responses for a GET request, but none is fresh, the client can use the ETag Option to perform a conditional request to the originating server. If a cached client response is still valid, the server can update the freshness of the cached response by including a Max-Age option in the response along with a code of 2.03 - Valid. If none of the cached responses is valid the server can respond with an updated representation of the resource and a response code of 2.05 - Content.

By default, caching is enabled in the CoAP protocol since the Max-Age Option defaults to a value of 60. Hence if the Max-Age option is not present in a cacheable response, then the response is considered not fresh after its age is greater than 60 s. If a server wishes to prevent caching, it shall explicitly include a Max-Age Option with a value of zero second.

Unlike HTTP, the cacheability of CoAP responses does not depend on the request method. Instead the cacheability of a response is dependent on the CoAP response code.

D.1.4 Method mapping

The primitives shall be mapped as follows.

Table D.1: CoAP Method Mapping

Primitive type	CoAP METHOD
xxxRetrieveRequestIndication	GET
xxxUpdateRequestIndication	PUT
xxxCreateRequestIndication	POST
xxxDeleteRequestIndication	DELETE
xxxExecRequestIndication	POST
xxxNotifyRequestIndication	POST (asynchronous notify) or Response to GET using CoAP observe mechanism [47]

Although CoAP allows CREATE to be implemented with a PUT, the present document shall not use this principle.

D.1.5 URI Options

The CoAP Uri-Path, Uri-Query and Proxy-Uri options shall be derived from the targetID primitive type attribute.

When addressing a remote resource on the mIa or dIa interface the targetID shall be used directly as the Proxy-Uri in CoAP, i.e. the local SCL acts as a CoAP proxy for the request.

When addressing a local resource, or when used on the mId interface, the query part of the targetID shall be used as the value for the Uri-Query CoAP option and the abs_path shall be used as the Uri-Path CoAP option.

For example, if the targetID has the following form:

- coap://NSCL.m2mSP.org:6123/discovery?searchString=example.

Then the CoAP options for a proxied retrieve would look like this:

- Proxy-Uri: NSCL.m2mSP.org:6123/discovery?searchString=example.

The CoAP options for a local (non-proxied) retrieve would look as follows (one Uri-Path Option per path token):

- Uri-Path: /*discovery*
- Uri-Query: searchString=example

Due to differential encoding rules for Options, the Options shall be incorporated into the CoAP message to send in the following order:

- Content-Type.
- Max-Age.
- Proxy-Uri.
- E-Tag.
- Location-Path.
- Location-Query.
- Uri-Path.
- Token.
- Accept.
- If-Match.
- Uri-Query.
- If-None-Match.

Proxy-Uri option shall take precedence over any of the Uri-Path or Uri-Query options (which shall not be included at the same time).

D.1.6 Blockwise Transfers

CoAP is based on datagram transports such as UDP, which limit the maximum size of resource representations that can be transferred without creating unreasonable levels of IP fragmentation. In addition, not all resource representations will fit into a single link layer packet of a constrained network (e.g. 127 byte MTU for 802.15.4) which may cause link layer fragmentation even if IP layer fragmentation is not required. Using fragmentation (either at the link layer or at the IP layer) to enable the transport of larger representations is possible up to the maximum size of the underlying datagram protocol (such as UDP), but the fragmentation/reassembly is a burden on the lower layers of resource constrained devices and can be more efficiently managed in the CoAP layer.

CoAP supports Block options to provide a minimal way to transfer larger representations in a block-wise fashion. Using these options, larger resource representations can be fragmented and reassembled by CoAP independently of the lower layers as well as the above application.

It is recommended that if the size of the resource representation exceeds the size of underlying network MTU then the CoAP Blockwise transfer function should be used.

To use the CoAP Blockwise transfer functionality, the CoAP Block1 option shall be used to define the size of the blocks used for requests and the CoAP Block2 option shall be used to define the size of the blocks used for responses (see [46] for further details). The supported block size values are in the range of 16 bytes to 1 024 bytes (in power of two increments).

D.2 Primitive mapping

The CoAP messages shall be mapped into primitives as follows.

Table D.2: CoAP Method Mapping

CoAP Method	Body content	Uri-Path	Primitive type
POST	Full <resourceType> representation	Parent resource for <resourceType>	<resourceType>createRequestIndication
GET	-	Resource of type <resourceType>	<resourceType>RetreiveRequestIndication
PUT	Full <resourceType> representation	Resource of type <resourceType>	<resourceType>UpdateRequestIndication
DELETE	-	Resource of type <resourceType>	<resourceType>DeleteRequestIndication
POST	Notify body reference to subscription child of <resourceType> and full <resourceType> representation	Any resource	<resourceType>NotifyRequestIndication
Partial addressing			
POST	Notify body reference to subscription child of <resourceType> and partial <resourceType> representation	Any resource	<resourceType>NotifyRequestIndication
POST	Partial <resourceType> representation	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication
GET	-	Resource of type <resourceType> with attribute accessor	<resourceType>RetreiveRequestIndication
DELETE	-	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication
PUT	Partial <resourceType> representation	Resource of type <resourceType> with attribute accessor	<resourceType>UpdateRequestIndication

D.2.1 Outgoing Request primitive to CoAP

- The primitive type shall be mapped to a CoAP Method according to the table in table D.1.
- The CoAP request-line shall be constructed using the selected CoAP method, selected options as described in clause D.1.5.
- For request addressing local resource or when using the mId, the Uri-Path shall be set according to clause D.1.5.

- If the request primitive indicated a resource representation, then the content shall be included in the proper representation in the message payload, including the correct Content-Type and/or in the CoAP Uri-Query option:
    - The CoAP payload shall be a sequence of XML description for either the user data part or the control parts of the resource representation, but typically not both since CoAP does not support multi-part/mixed MIME (RFC 2046 [15]) encoded payloads.
    - For cases where a resource representation consists of both a control part and a user data part of different Content-Type, several options exist as described in clause D.1.1.
- The content payload shall end the mapped CoAP request message. According to primitive modelling, the content payload shall be the sequence of descriptions of control and user data parts with the same Content-Type.
- If the request is a RETRIEVE request and contains *filterCriteria* or *contentInstancesFilterCriteria* as an primitive parameter, then each element in this filterCriteria shall be mapped as a query parameter in the requestURI, where the name of the query parameter is the element name of the *filterCriteria* or *contentInstancesFilterCriteria* attribute, and the value shall be an URI-encoded representation of the corresponding value in as it would appear in the XML representation. The same query parameter can occur multiple times if the maxOccurs parameter in the XML representation is more than one.
  For example, the filterCriteria:

```
        <filterCriteria>
           <searchString>test1</searchString>
           <searchString>test2</searchString>
        </filterCriteria>
    is mapped as
         ?searchString=test1&searchString=test2
```

  The query parameters are mapped to the CoAP Uri-Query options as specified in clause D.1.5.

D.2.2 Incoming CoAP response to responseConfirm primitive

The CoAP response shall be correlated with the outgoing request. The issuer and receiver are the reverse of the corresponding request primitive.

If the CoAP response code is in the range 2.01 to 2.05, then the response shall be considered as a successful response. Any control or user data is mapped to the corresponding resource representation or response data structure (for virtual resources).

The CoAP statusCodes for successful operations shall be mapped to ETSI statusCodes as follows:

Table D.3: CoAP statusCodes mapping to ETSI statusCodes (successful cases)

CoAP statusCode	StatusCode in request primitive
2.02, 2.03, 2.04, 2.05	"STATUS_OK"
2.01	"STATUS_CREATED"

If the CoAP response code is in the range of 4.00 to 4.15 or 5.00 to 5.05, then this is an error and the body shall be interpreted as additional information about the error.

The error response shall contain a body, and the body shall be an *errorInfo* primitive attribute containing a *statusCode* element and an optional additionalInfo element. it shall be mapped to the *errorInfo* primitive attribute in the unsuccessful responseConfirm.

D.2.3 Incoming CoAP request to Request primitive

If the CoAP Proxy-Uri includes an absolute URI and the host and port part of the URI do not match the host and port on which the request was received, then receiving SCL shall check if it has a secure association with the host and port indicated in the URI, and if so, extracts the host and port part of the URI. The receiving SCL shall forward the request over that secure channel, with a Uri-Path option matching the path portion of the original Proxy-Uri option. If no secure channel is established with the indicated host and port, the request shall be rejected with 4.01 Unauthorized.

The primitive shall be derived from the combination of CoAP method and the type of the addressed resource as taken from the Uri-Path option.

D.2.4 Outgoing responseConfirm primitive to CoAP response

The primitive shall be correlated to the corresponding request primitive and the following rules shall be followed:

- The CoAP response shall be correlated to the CoAP request corresponding to the request primitive.
- The statusCode shall be the status code from the primitive.
- If the response primitive has any resource representation, this shall be transported in the payload of the response.
- The CoAP message shall be sent to issuer over the data transfer function.
- The statusCode of the response for successful and unsuccessful responseConfirm shall be set according to the table below. For the successful operations, STATUS_OK shall be mapped to the CoAP statusCode corresponding to the specific operation.

Table D.4: ETSI statusCodes mapping to CoAP statusCodes (successful cases)

StatusCode responseConfirm primitive	StatusCode in CoAP
"STATUS_OK"	2.02, 2.03, 2.04, 2.05
"STATUS_CREATED"	2.01
"STATUS_ACCEPTED"	ACK

Table D.5: ETSI statusCodes mapping to CoAP statusCodes (error cases)

StatusCode in errorInfo	StatusCode in CoAP
"STATUS_BAD_REQUEST"	4.00
"PERMISSION_DENIED"	4.01
"STATUS_FORBIDDEN"	4.03
"STATUS_NOT_FOUND"	4.04
"STATUS_METHOD_NOT_ALLOWED"	4.05
"STATUS_NOT_ACCEPTABLE"	4.00
"STATUS_REQUEST_TIMEOUT"	4.00
"STATUS_CONFLICT"	4.00
"STATUS_UNSUPPORTED_MEDIA_TYPE"	4.15
"STATUS_INTERNAL_SERVER_ERROR"	5.00
"STATUS_NOT_IMPLEMENTED"	5.01
"STATUS_BAD_GATEWAY"	5.02
"STATUS_SERVICE_UNAVAILABLE"	5.03
"STATUS_GATEWAY_TIMEOUT"	5.04
"STATUS_EXPIRED"	4.00
"STATUS_DELETED"	4.04

Detailed descriptions of the CoAP response codes used in tables D.4 and D.5 are given in table D.6.

Table D.6: Definitions of CoAP statusCodes [45]

Code No.	Response Code Name	Description
2.01	Created	Similar to HTTP 201 "Created". Used in response to POST and PUT requests.
2.02	Deleted	Similar to HTTP 204 "No Content". Used in response to DELETE. requests.
2.03	Valid	Related to HTTP 304 "Not Modified", but only used to indicate that the response identified by the entity-tag identified by the included ETag Option is valid. The response SHALL include an ETag Option.
2.04	Changed	Similar to HTTP 204 "No Content". Used in response to POST and PUT requests
2.05	Content	Similar HTTP 200 "OK" but only used in response to GET requests. The payload returned with the response is a representation of the target resource.
4.00	Bad Request	Similar to HTTP 400 "Bad Request".
4.01	Unauthorized	The client/issuer is not authorized to perform the requested action.
4.02	Bad Option	The request could not be understood by the server due to one or more unrecognized or malformed critical options.
4.03	Forbidden	Similar to HTTP 403 "Forbidden".
4.04	Not Found	Similar to HTTP 404 "Not Found".
4.05	Method Not Allowed	Similar to HTTP 405 "Method Not Allowed", but with no parallel to the "Allow" header field.
4.12	Precondition Failed	Similar to HTTP 412 "Precondition Failed".
4.13	Request Entity Too Large	Similar to HTTP 413 "Request Entity Too Large".
4.15	Unsupported Media Type	Similar to HTTP 415 "Unsupported Media Type".
5.00	Internal Server Error	Similar to HTTP 500 "Internal Server Error".
5.01	Not Implemented	Similar to HTTP 501 "Not Implemented".
5.02	Bad Gateway	Similar to HTTP 502 "Bad Gateway".
5.03	Service Unavailable	Similar to HTTP 503 "Service Unavailable", but using the Max-Age Option in place of the "Retry-After" header field.
5.04	Gateway Timeout	Similar to HTTP 504 "Gateway Timeout".
5.05	Proxying Not Supported	The server is unable or unwilling to act as a proxy for the URI specified in the Proxy-Uri Option.

D.2.5 Mapping of primitive attributes

The primitive attributes shall be mapped as follows.

Table D.7: Mapping of primitive attributes to CoAP

Attribute Name	CoAP Mapping
Issuer	The issuer is identified by the DTLS session that is setup to exchange communication between the issuer and receiver. If DTLS is not used, the UDP source in the datagram can be used.
Receiver	The receiver is identified by the DTLS session that is setup to exchange communication between the issuer and receiver. If DTLS is not used, the UDP destination in the datagram can be used. Note that in case of forwarding, two separate DTLS sessions are used.
RequestingEntity	RequestingEntity shall be included in CoAP uri-query header option in the format of the type AnyURI. ?RequestingEntity=<value> Where <value> is a placeholder for an URI in the format of AnyURI as defined in clause 11.2.
targetID	This is the Proxy-Uri CoAP option when addressing a remote resource. When addressing a local resource, this is the value of the Uri-Path option of the CoAP request. For example, for a remote request on the dIa interface, the requestURI might look like this: GET Proxy-Uri: coap://networkSCL.m2mprovider.org/applications/testApp. On the mId it might look like this: GET Uri-Path: /applications/testApp
TRPDT	This attribute is included in the CoAP uri-query header option: ? x-etsi-trpdt=<value> Where <value> is a placeholder for a string in the format of the TrpdtType as defined in clause 11.4.
RCAT	This attribute is included in the CoAP uri-query header option: ?x-etsi-rcat=<value> Where <value> is a placeholder for a string according to the definition of the enumerated RcatType in clause 11.3.

When the receiver is a proxy, the query string shall be used with a Proxy-Uri option.

D.2.6 Partial addressing

In CoAP a resource is normally manipulated in its entirety. i.e. to provide a new representation, the old resource representation is replaced. Some modifications, e.g. additions are usually done with POST. However, CoAP does not define a generic mechanism for modifying the content within a resource in an idempotent and safe way (POST is not idempotent, unless additional measures are taken; PUT is idempotent but not safe).

An alternative used in the present document is to allow the addressing of XML data inside the CoAP resource using principles similar to the spirit of REST, i.e. by allowing REST primitives to operate on embedded XML or JSON data. This is an approach popularized by XCAP [i.4], which does allow arbitrary xpath expressions to manipulate parts of the representation of a resource. The approach taken here is very similar. I.e. the parts of the resource are identified using normal URIs, where the components correspond to the names of the attributes, or ids of the attributes when selecting a sub-attribute in a collection.

In this way attributes can be removed (DELETE, if allowed), updated (PUT), created (PUT), values can be added to collections (POST). And of course the value of a single attribute can be obtained (GET).

The partial addressed attributes share the ETags and Max-Ages of their containing CoAP resource, i.e. it can be seen as a filtering mechanism, to return or manipulate only a part of the data. However, this also means that conditional requests are based on the complete resource.

For example, suppose that resource {X} has two attributes called y and z.

Using partial addressing somebody can read only attribute y from resource {X}, i.e. using GET {X}/y

This will return only the value of attribute y of resource {X} and an e-tag, say, 42.
Using partial addressing somebody now writes a new value for attribute z; PUT {X}/z with a body containing the new value of z. This returns a 2. 04 Changed response and an e-tag of 12.
The e-tag applies to the entire resource, so GET {X}/y will return the old value of y, even though this (filtered) part of the resource has not been changed.

D.3 Semi-asynchronous and asynchronous communication

Since CoAP already supports asynchronous communication, semi-asynchronous communication will not be used.

Annex E (normative): Mapping of Management Objects

E.1 Introduction

This annex contains all information on how to map specialized versions of the generic <mgmtObj> resource to their equivalent managed object as defined in OMA DM or BBF TR-069 [49] protocols.

All ETSI M2M defined <mgmtObj> resource instances shall be mapped to their equivalent managed object as defined in OMA DM or BBF TR-069 [49] protocols as specified in this clause.

E.2 Data Types only used for attributes in <mgmtObj> resources

E.2.1 Purpose

This defines the data types that shall be used for attributes in <mgmtObj>. These types are based on the data types definitions in clause 11.

E.2.2 Complex data types

RankedAnList

Ranked list of Tokens that indicate preferred access networks.

Table E.1

Name	Type	Description
accessNetwork	Stoken [1..unbounded]	A list of access network names where each item is a token that identifies a specific access network in line with the anName attribute of an <anpPolicy> resource. The order of the tokens is significant: The first token in the list represents the most preferred access network. The last token in the list indicates the least preferred access network.

AbsTimeSpan

Defines absolute time spans.

Table E.2

Name	Type	Description
startTime	DateTime	Start of time span in absolute time
endTime	DateTime	End of time span in absolute time. Constraint: endTime > startTime

SchedItem

Defines allowed time spans for scheduling decisions. Items in the 'schedule' list define allowed time spans as recurring time spans. Items of the list in 'absTimeSpans' represent the absolute time spans in which the recurrence of allowed time spans defined in 'schedules' is constrained. If 'absTimeSpans' is empty, no constraints apply to the allowed time spans defined in 'schedules'. If 'schedule' is empty, all time spans defined in 'absTimeSpans' are allowed. At least one of 'absTimeSpans' and 'schedule' shall not be empty. All items in the list 'absTimeSpan' shall use the same time zone. If 'absTimeSpans' is not present or does not define a specific time zone, UTC shall be assumed for all time spans defined in that instance of SchedItem.

Table E.3

Name	Type	Description
absTimeSpan	AbsTimeSpan [0..unbounded]	List of allowed absolute time spans to be associated with the RCAT value
schedule	Schedule	List of allowed time spans expressed as a Schedule to be associated with the RCAT value

RcatSchedule

Associations between allowed time spans and a specific RCAT value.

Table E.4

Name	Type	Description
rcatValue	rcatType	The RCAT value with which to associate allowed time spans.
schedItem	SchedItem [1..unbounded]	A list of items of type RcatSchedItem defining allowed time spans for the given RCAT value. Scheduling of traffic of the given RCAT category is allowed during all time spans in this list.

RcatSchedList

List of associations between allowed time spans and non-overlapping RCAT values.

Table E.5

Name	Type	Description
rcatSchedule	RcatSchedule [1..maximum number of RCAT values]	A list of items where each item associates allowed time spans with a specific RCAT value. The item-specific RCAT values shall not be overlapping, i.e. at most one item in this list can exist with a given value for RCAT.

NOTE: 'maximum number of RCAT values' is equal to 8.

BlockItem

Associates a duration for which an access network shall not be used when a number of consecutively failed access attempts have been executed.

Table E.6

Name	Type	Description
failedAttempts	Long	Defines the number of consecutively failed access attempts for which the duration below should be used to block the next access attempt.
blockDuration	Duration	Defines the duration for which to block further access attempts

BlockList

List of durations to block access attempts as a function of the number of consecutively failed attempts.

Table E.7

Name	Type	Description
blockItem	BlockItem [1..unbounded]	A list of items where each item defines a number of consecutively failed access attempts and a duration that should be used to block the next access attempt. The item with the largest number of consecutively failed attempts that is smaller or equal to the actual number of consecutively failed attempts will apply in the SAF handling

ActionStatus

Status of the operation performed on all the action related subresources in clause E.3.

Table E.8

Name	Type	Description
action	AnyURI	Reference to the action (represented by a resource attribute) being performed
progress	Short	Provides an indication of the progress of the action
finalStatus	FinalStatus	The final status of the action

OccuredEvents

Indicates the last occurences of an event.

Table E.9

Name	Type	Description
currentIndex	String	Indicates the rank of the last occurred event in the table of timeStamps
trapEventTimeStamp	DateTime[1..100]	a circular buffer of timeStamps of the last occurred events

NameValuePairItem

Defines one of management parameters for a M2M Area Network.

Table E.9a

Name	Type	Description
name	String	Contains the name of one specific parameter used for describing a M2M Area Network
value	String	Contains the value of the parameter represented by "name"

AreaNwkTypeInfoSet

A set of parameters specific to the type of M2M Area Network denoted by the "areaNwkType" attribute.

Table E.9b

Name	Type	Description
areaNwkTypeItem	NameValuePairItem[0..unbounded]	A set of Items that contain parameters describing a M2M Area Network. Each parameter is characterized by a name and a value. Each Name shall be unique within the AreaNwkTypeInfoSet.

E.2.3 Enumeration data types

Table E.10

FinalStatus	Predefined values: "SUCCESS" "FAILURE"	Enumeration of the status of a management action defined in some ETSI M2M mgmtObjs. For example, ReRegistration in etsiSclMo
AreaNwkType	Predefined values: "6LOWPAN" "6LOWPAN-BLUETOOTH" "6LOWPAN-WIFI" "6LOWPAN-PLC" "IPV4-WPAN" "IPV4-BLUETOOTH" "IPV4-WIFI" "IPV4-PLC" "ZIGBEE"	Enumeration of the type of an M2M area network
AreaNwkStatus	Predefined values: "ASLEEP" "AWAKE"	Enumeration of the status of an M2M Device in an M2M Area Network

E.3 Mapping for Management Objects to their equivalents in OMA DM and BBF

E.3.1 References and general mapping assumptions

The OMA-DM Management Objects that are used in this annex are described in the following references:

- DevInfo, DevDetail Management Objects are described in [50].
- Device Capability Management Object described in [51].
- Battery, Memory, Diagnostic, and Restart Management Objects described in [52].
- Event Trap Management Object described in [54] and [55].
- Firmware Update Management Object described in [56].
- Software Component Management Object described in [57].
- Factory Reset Management Object described in [53].

The BBF TR-069 data model is described in the references [58], [59] and [60].

One mapping table per ETSI-defined resource is specified. In each mapping table, only the MO-specific attributes are mentioned. The common attributes (expirationTime, accessRightID, searchStrings, creationTime, lastModifiedTime, moID, originalMO, description) do not play a role in the mapping and they are not mentioned in tables E.11 to E.22.

The RPC method or DM command that is used in the BBF/OMA management protocols is mentioned for each ETSI-defined resource.

When one attribute coming from a M2M Application can be mapped on several BBF or OMA parameters, the mapping rule is defined in tables E.11 to E.22

E.3.2 Resource etsiSclMo

Resource etsiSclMo shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.11.

Table E.11: etsiSclMo resource tree mapping to OMA DM and BBF TR-069 [49]

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in updateReq	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	regTargetNsclList	M	O	M	AnyURIList	<x>/RegTargetNsclList	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.Reregistration.RegTargetNSCLList	string (comma separated list of strings)
	regExpirationDuration	O	O	M	Duration	<x>/RegExpirationDuration	int	Device.ETSIM2M.SCL.{i}.Reregistration.RegExpirationDuration	long
	regAccessRightID	O	O	M	AnyURI	<x>/RegAccessRightID	chr	Device.ETSIM2M.SCL.{i}.Reregistration.RegAccessRightID	string
	regSearchStrings	O	O	M	SearchStrings	<x>/RegSearchStrings	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.Reregistration.RegSearchStrings	string (comma separated list of strings)
	announcedToSclList	M	O	M	AnyURIList	<x>/AnnouncedToSclList	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.AnnouncedToSCLList	string (comma separated list of strings)
	maxNumberOfDiscovRecords	O	O	M	Long	<x>/MaxNumberOfDiscovRecords	int	Device.ETSIM2M.SCL.{i}.Discovery.MaxNumberOfDiscovRecords	unsignedInt
	maxSizeOfDiscovAnswer	O	O	M	Long	<x>/MaxSizeOfDiscovAnswer	int	Device.ETSIM2M.SCL.{i}.Discovery.MaxSizeOfDiscovAnswer	unsignedInt
	actionStatus	NP	NP	M	ActionStatus	N/A	N/A	N/A	N/A
	actionStatus/progress	NP	NP	M	Short	<x>/OperationStatus/Progress	int	Device.ETSIM2M.SCL.{i}.Reregistration.ActionStatus.Progress	unsignedInt[0:100]
	actionStatus/finalStatus	NP	NP	M	FinalStatus	<x>/OperationStatus/FinalStatus	chr	Device.ETSIM2M.SCL.{i}.Reregistration.ActionStatus.FinalStatus	string
sclMoAction	reRegistration	O	O	M	AnyURI	<x>/Operations/ReRegistration	null	Device.ETSIM2M.SCL.{i}.Reregistration.Reregistration	boolean
	deRegistration	O	O	M	AnyURI	<x>/Operations/DeRegistration	null	Device.ETSIM2M.SCL.{i}.Enable	boolean
<safPolicySet>	policyScope	M	O	M	AnyURIList	<x>/SafPolicySets/<x>/PolicyScope	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.PolicyScope	string (comma separated list of strings)
<safPolicySet>/m2mSpPolicy	defaultRcatValue	M	O	M	RcatType	<x>/SafPolicySets/<x>/M2mSpPolicy/DefaultRcatValue	chr	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.DefaultRCATValue	string
<safPolicySet>/m2mSpPolicy/<rcatParamList >	rcatValue	M	O	M	RcatType	<x>/SafPolicySets/<x>/M2mSpPolicy/RcatParamList/<x>/RcatValue	chr	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.RequestCategory.{i].RCAT	string
	defaultTrpdtValue	M	O	M	TrpdtType	<x>/SafPolicySets/<x>/M2mSpPolicy/RcatParamList /<x>/DefaultTrpdtValue	int	N/A	N/A
	defaultTrpdtValue/tolerableDelay	M	O	M	Duration	N/A	N/A	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.RequestCategory.{i].TolerableDelay	int

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in createReq	M/O in updateReq	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
	maxPendReqs	O	O	M	Long	<x>/SafPolicySets/<x>/M2mSpPolicy/RcatParamList/<x>/MaxPendReqs	int	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.RequestCategory.{i].Thresh	unsignedInt
	maxPendData	O	O	M	MemorySize	<x>/SafPolicySets/<x>/M2mSpPolicy/RcatParamList/<x>/MaxPendData	chr	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.RequestCategory.{i].Mem	string(16)
	rankedAnList	O	O	M	RankedAnList	<x>/SafPolicySets/<x>/M2mSpPolicy/RcatParamList/<x>/RankedAnList	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.M2MSPPolicy.RequestCategory.{i].RankedANList	string (comma separated list of strings)
<safPolicySet>/<anpPolicy>	anName	M	O	M	Stoken	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/AnName	chr	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}.ANName	string
	rcatSchedList	M	O	M	RcatSchedList	N/A	N/A	N/A	N/A
	rcatSchedList/<rcatSchedule>/rcatValue	M	O	M	RcatType	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/RcatValue	chr	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. RequestCategory.{i].RCAT	string
	rcatSchedList/<rcatSchedule>/<schedItem>/schedule	M	O	M	Schedule	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/SchedItems/<x>/Schedule	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. RequestCategory.{i].Schedule.{i}.Schedules	string (comma separated list of strings)
	rcatSchedList/<rcatSchedule>/<schedItem>/<absTimeSpan>/startTime	M	O	M	DateTime	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/SchedItems/<x>/AbsTimeSpans/<x>/StartDate <x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/SchedItems/<x>/AbsTimeSpans/<x>/StartTime	date time	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. RequestCategory.{i].Schedule.{i}.AbsTimeSpan.{i}.StartTime	dateTime
	rcatSchedList/<rcatSchedule>/<schedItem>/<absTimeSpan>/endTime	M	O	M	DateTime	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/SchedItems/<x>/AbsTimeSpans/<x>/EndDate <x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/RcatSchedList/<x>/SchedItems/<x>/AbsTimeSpans/<x>/EndTime	date time	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. RequestCategory.{i].Schedule.{i}.AbsTimeSpan.{i}.EndTime	dateTime
	blockPeriods	O	O	M	BlockList	N/A	N/A	N/A	N/A
	blockPeriods/<blockItem>/failedAttempts	O	O	M	Long	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/BlockPeriods/<x>/FailedAttempts	Int	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. BlockPeriod.{i].FailedAttempts	unsignedInt
	blockPeriods/<blockItem>/blockDuration	O	O	M	Duration	<x>/SafPolicySets/<x>/M2mAnpPolicySets/<x>/BlockPeriods/<x>/BlockDuration	int	Device.ETSIM2M.SCL.{i}.SAFPolicySet.{i}.ANPPolicy.{i}. BlockPeriod.{i].BlockDuration	int
NOTE: The term N/A in this table refers to the current state of analysis.									

E.3.3 Resource etsiDeviceInfo

Resource etsiDeviceInfo shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.12.

etsiDeviceInfo is used by a M2M Application for getting information from the M2M Device or M2M Gateway.

In the OMA-DM case, the information is got by using the OMA-DM Get command.

For the BBF TR-069 case, the information is got by using the GetParameterValues RPC method.

Table E.12: etsiDeviceInfo resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in updateReq	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	deviceLabel	NP	N/A	M	String	DevInfo/DevId	chr	Device.DeviceInfo.SerialNumber	String(64)
	manufacturer	NP	N/A	M	String	DevInfo/Man	chr	Device.DeviceInfo.Manufacturer	String(64)
	model	NP	N/A	M	String	DevInfo/Mod	chr	Device.DeviceInfo.ModelName	String(64)
	deviceType	NP	N/A	M	String	DevDetail/DevType	chr	Device.DeviceInfo.ProductClass	String(64)
	firmwareVersion	NP	N/A	M	String	DevDetail/FwV	chr	Device.DeviceInfo.SoftwareVersion	String(64)
	softwareVersion	NP	N/A	M	String	DevDetail/SwV	chr	Device.DeviceInfo.SoftwareVersion	String(64)
	hardwareVersion	NP	N/A	M	String	DevDetail/HwV	chr	Device.DeviceInfo.HardwareVersion	String(64)

The "deviceLabel" attribute does not intend to be used by a M2M Application or any other entity as a globally unique device identity. This deviceLabel is the information as assigned by the Manufacturer and, as such, its uniqueness may be global or only valid within a certain domain (e.g. vendor-wise).

Due to the current non-distinction between firmware and software in the BBF-TR-069 data model, the TR 069 SoftwareVersion is mapped on both the ETSI softwareVersion and ETSI firmwareVersion. (In TR-069 the SoftwareVersion may also be referred as "the version of the overall CPE firmware").

E.3.4 Resource etsiDeviceCapability

Resource etsiDeviceCapability shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.13.

For the BBF TR-069 case, the SetParameterValues RPC method is used for setting the parameters.

In the OMA-DM case, when a M2M Application sends an UPDATE request on the sub-resource capabilityAction/enable (or /disable), this corresponds to an OMA-DM Exec command for enabling (/disabling) the Device Capability to transfer the Device Capability from Disabled (Enabled) State to Enabled (/Disabled) State.

Table E.13: etsiDeviceCapability resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
capabilityInstance	CapabilityName can be:								
	Display,	NP	N/A	M	String	Device Capability/Property	chr	N/A	
	Camera					Device Capability/Property	chr	N/A	
	Speaker					Device Capability/Property	chr	N/A	
	USB					Device Capability/Property	chr	Device.USB.Port.(i).Name	String(64)
	SerialPort					Device Capability/Property	chr	N/A	
	GPS					Device Capability/Property	chr	N/A	

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
	Bluetooth					Device Capability/Property	chr	N/A	
	Infrared					Device Capability/Property	chr	N/A	
	2G					Device Capability/Property	chr	N/A	
	3G					Device Capability/Property	chr	N/A	
	4G					Device Capability/Property	chr	N/A	
	WLAN					Device Capability/Property	chr	No Name even when WLAN is supported	
	Media Server					N/A		No Name even when a Media Server is supported through UpnP	
	Smart Card Reader					N/A		Device.SmartCardReaders.SmartCardReader.{i}.Name	string(256)
	WiFi					N/A		Device.WiFi.Radio.{i}.Name	string(64)
	HomePlug					N/A		Device.HomePlug.Interface.{i}.Name	string(64)
	MoCA					N/A		Device.MoCA.Interface.{i}.Name	string(64)
	UPA					N/A		Device.UPA.Interface.{i}.Name	string(64)
	attached	NP	N/A	O	Boolean	Device Capability/Attached	bool	N/A	
	capabilityActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the capabilityActionStatus is set to 0% as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100% and the final state becomes valid.(success, failure). Note: this means that during the Device Capability MO handling by NREM, the parameter /DCMOConfig/NotifyUser has to be used and set to "true" (in order to be notified of the status of the operation)		No direct mapping with a BBF element. The progress indicator of the capabilityActionStatus is set to 0% as soon as the action is taken into accountby NREM. After completion of the action, the progress indicator is set to 100% and the final state becomes valid.(success, failure).	
CapabilityInstance /capabilityAction	devCapEnable	NP	NP	O	AnyURI	Device Capability/Operations /Enable	null	For USB: Device.USB.Interface.(i).Enable	boolean
								For WLAN: Device.UpnP.Device.UPnPWLANAccessPoint	boolean
								For Media Server: Device.UpnP.Device.UPnPMediaServer	boolean
								For HomePlug interface: Device.HomePlug.Interface.{i}.Enable	boolean
								For MoCA interface: Device.MoCA.Interface.{i}.Enable	boolean
								For SmartCardReader: Device.SmartCardReaders.SmartCardReader.{i}.Enable	boolean
								For WiFi interface: Device.WiFi.Radio.{i}.Enable	boolean

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
								For UPA interface: Device.UPA.Interface.{i}.Enable	boolean
	devCapDisable	NP	NP	O	AnyURI	Device Capability/Operations/Disable	null	For USB: Device.USB.Interface.(i).Enable	boolean
								For WLAN: Device.UpnP.Device.UPnPWLANAccessPoint	boolean
								For Media Server: Device.UpnP.Device.UPnPMediaServer	boolean
								For HomePlug interface: Device.HomePlug.Interface.{i}.Enable	boolean
								For MoCA interface: Device.MoCA.Interface.{i}.Enable	boolean
								For SmartCardReader: Device.SmartCardReaders.SmartCardReader.{i}.Enable	boolean
								For WiFi interface: Device.WiFi.Radio.{i}.Enable	boolean
								For UPA interface: Device.UPA.Interface.{i}.Enable	boolean

When 2G or 3G or 4G is used as capabilityName by the Application, NREM has to map this value to the right one among ["EDGE", "GPRS", "UMTS", "HSDPA", "HSUPA", "HSPA+", "LTE", "PACKET", "WCDMA", "CDMA", "TD-SCDMA"].

E.3.5 Resource etsiBattery

Resource etsiBattery shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.14.

etsiBattery is used by a M2M Application for getting some battery information from the M2M Device or M2M Gateway. A RETRIEVE request from a M2M Application on the etsiBattery resource will trigger a Get OMA-DM command.

Table E.14: etsiBattery resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	standbyTime	NP	N/A	M	Long	DiagMonFcts/Battery/DiagMonData/<x>/BatteryStandbyTime	int	N/A	N/A
batteryInstance	battLevel	NP	N/A	M	Long	DiagMonFcts/Battery/DiagMonData/<x>/BatteryLevel	int	N/A	N/A
	battStatus	NP	N/A	M	Long	DiagMonFcts/Battery/DiagMonData/<x>/BatteryStatus	int	N/A	N/A

E.3.6 Resource etsiMemory

Resource etsiMemory shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.15.

A RETRIEVE request from a M2M Application on the etsiBattery resource will trigger:

- In the OMA case, a Get OMA-DM command.
- In the BBF TR-069 case, a GetParameterValues RPC method.

Table E.15: etsiMemory resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in updateReq	M/O in create Req	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	ramAvailable	NP	N/A	M	Long	DiagMonFcts/Memory/DiagMonData/RAMAvail	Int	Device.DeviceInfo.MemoryStatus.Free	unsignedint
	ramTotal	NP	N/A	M	Long	DiagMonFcts/Memory/DiagMonData/RAMTotal	int	Device.DeviceInfo.MemoryStatus.Total	unsignedint

E.3.7 Resource etsiTrapEvent

Resource etsiTrapEvent shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.16.

Table E.16: etsiTrapEvent resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
trapInstance	trapId	M	M	M	String	DiagMonFcts/Trap Event Logging Function/ DiagMonConfig/ConfigParms/TrapEvents/ <x>/Ref	chr	"X <VENDOR> <EVENT>" as defined in [55]	
	EventOccured	NP	NP	O	OccuredEvents	The content of this parameter is coming from DiagMonFcts/Trap Event Logging Function/DiagMonData/Log	xml	The content of this parameter is coming from the content of the "Inform Request" relative to the Vendor-specific event	
	trapActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the trapActionStatus is set to 0% as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(started, stopped, failure).		No direct mapping with a BBF element. The progress indicator of the trapActionStatus is set to 0 % as soon as the action is taken into accountby NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(started, stopped, failure).	
TrapInstance /trapEventAction	TrapEventEnable	NP	NP	O	AnyURI	DiagMonFcts/Trap Event Logging Function /Operations/Start	null	This parameter doesn't exist in the BBF approach. When "enable" is requested by a NA that corresponds to a TR-069 device, then NREM has to use the "Inform Request" messages coming from the Device or Gateway via the TR-069 protocol in order to begin to report the event occurrence to the NA.	
	trapEventDisable	NP	NP	O	AnyURI	DiagMonFcts/Trap Event Logging Function /Operations/Stop	null	When a NA requests "disable", then NREM stops to report to the NA the content of the "Inform Request" messages received from the M2M Device or M2M Gateway.	

The TrapId has to correspond to either one of the identifier used by OMA for identifying a specific trap event or a Vendor-specific event as defined by BBF in TR-069 [49].

During the OMA's MO handling by NREM, the parameter "/DiagMonConfig/ConfigParms/Type" has to be set to "circular", and the parameter "/DiagMonConfig/ConfigParms/Info/TS" has to be set to "True" (in order to have the timestamps recorded).

When a trap is enabled, NREM updates the circular buffer of the etsiTrapEvent resource with the timestamps of the event occurrences. For doing that, NREM receives Inform Request from the M2M Device or M2M Gateway in the case of BBF-TR-069. In the case of OMA, NREM has to uses the OMA-DM Get command in order to periodically get the log file stored in the M2M device or M2M Gateway. The time stamps are then extracted from this file and used for updating the etsiTrapEvent resource.

E.3.8 Resource etsiPerformanceLog

Resource etsiPerformanceLog shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.17.

In the BBF TR-069 case, the SetParameterValues RPC method is used for setting the value "Requested".

In the OMA case, the log are started or stopped by using the Exec OMA-DM command.

Table E.17: etsiPerformanceLog resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in createReq	M/O in updateReq	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	LogTypeId (systLog or appLog)	M	M	M	String	N/A		N/A	N/A
	logData	NP	NP	O	xml	DiagMonFcts/Panic Logs/ <x>/DiagMonData/PanicLog (when logTypeId=systLog)	xml	Device.SelfTestDiagnostics.Results (Independently on the value of logTypeId, systLog or appLog)	String(1024)
					xml	DiagMonFcts/Application Monitoring Log/<x>/ DiagMonData/AppMonLog (when logTypeId=appLog)	xml		
	perfoLogActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the perfoLogActionStatus is set to 0 % as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(log_is_started, log_is_stopped, failed_to_start_log, failed_to_stop_log)		No direct mapping with a BBF element. The progress indicator of the perfoLogActionStatus is set to 0 % as soon as the action is taken into accountby NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(log_is_started, failed_to_start_log)	
perfLogAction	perfLogStart	NP	NP	O	AnyURI	DiagMonFcts/Panic Logs/ <x>/Operations/Start (when logTypeId=systLog)	Null	Device.SelfTestDiagnostics.DiagnosticsState = "Requested" (Independently on the value of logTypeId, systLog or appLog)	String
					AnyURI	DiagMonFcts/Application Monitoring Log/<x>/Operations/Start (when logTypeId=appLog)	Null		
	perfLogStop	NP	NP	O	AnyURI	DiagMonFcts/Panic Logs/ <x>/Operations/Stop (when logTypeId=systLog)	Null	Stop has no effect for a TR-069 device	N/A
					AnyURI	DiagMonFcts/Application Monitoring Log/<x>/Operations/Stop (when logTypeId=appLog)	null		

In the OMA case, two log files may be created depending on the requested input (systLog or appLog). The format of these files is not standardized by OMA, but the data contained in these files are encapsulated in xml format. For the etsiPerformanceLog resource, NREM will concatenate the two set of data in a single xml format. (with a clear separation by using tags sysLog or appLog).

E.3.9 Resource etsiFirmware

Resource etsiFirmware shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.18.

In the BBF TR-069 case the "Download" RPC method is used for downloading then installing a new firmware. The "Download" RPC is used with the "FileType" argument = 1 (Firmware Upgrade Image).

In the OMA case, the operations of download and update are performed by using the Exec OMA-DM command.

Table E.18: etsiFirmware resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
fwInstance	FirmwareName	M	NP	M	String restricted to 256	Firmware Update/<x>/PkgName	chr	"TargetFileName" argument of the "Download" method	String(256)
	FirmwareVersion	M	NP	M	String	Firmware Update/<x>/PkgVersion	chr	N/A	
	FirmwareURL	M	NP	M	String restricted to 256	Firmware Update/<x>/Download /PkgURL or Firmware Update/<x>/Download AndUpdate/PkgURL (depending on the requested action)	chr	"URL" argument of the "Download" method	String(256)
	fwActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the fwActionStatus is set to 0 % as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(download_failed or download_complete when the action is "download", update_failed or update_successful when the action is "update" or "downloadAndUpdate" , remove_failed, or remove_successful when the action is "remove")		No direct mapping with a BBF element. The progress indicator of the fwActionStatus is set to 0% as soon as the action is taken into accountby NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(download_failed or download_complete). In this case, the value "download_complete" means that the firmware file was downloaded and successfully applied.	
fwInstance /firmwareAction	FwDownload	NP	NP	O	AnyURI	Firmware Update/<x>/Download	No datatype , it is a node	N/A. A "download" action is not allowed for a TR069 device. An "unauthorized operation" will be returned to a NA that would request a "download".	
	FwUpdate	NP	NP	O	AnyURI	Firmware Update/<x>/Update	No datatype , it is a node	N/A. An "update" action is not allowed for a TR069 device. An "unauthorized operation" will be returned to a NA that would request an "update".	
	fwDownloadAndUpdate	NP	NP	O	AnyURI	Firmware Update/<x>/Download AndUpdate	No datatype , it is a node	Use of the "Download" RPC with the "FileType" argument = 1 (Firmware Upgrade Image)	
	fwRemove	NP	NP	O	AnyURI				

In the BBF TR-069 case, if authentication is required with the file server, NREM is supposed to know the Username/Password to be provided to the M2M device or M2M Gateway in the Download method.

E.3.10 Resource etsiSoftware

Resource etsiSoftware shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.19.

In the BBF TR-069 case, the change in the state of a piece of software are performed by using the ChangeDUState RPC method.

In the OMA case, the operations on a piece of software are performed by using an Exec OMA-DM command.

Table E.19: etsiSoftware resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
swInstance	softwareName	M	NP	M	String restricted to 64	SoftwareComponent/<x>/Download/<x>/Name ((when the action is "download" or "downloadAndInstall")	chr	Device.SoftwareModules.DeploymentUnit.{i}.Name	String(64)
					String restricted to 64	SoftwareComponent/<x>/ Inventory/Delivered/<x>/Name (when the action is "install" or "remove")	chr		
					String restricted to 64	SoftwareComponent/<x>/ Deployed/<x>/Name (when the action is "activate" or "deactivate")	chr		
	softwareVersion	M	NP	M	String restricted to 32	SoftwareComponent/<x>/ Inventory/ Deployed/<x>/Version (when the action is "activate", "deactivate" or "remove")	chr	Device.SoftwareModules.DeploymentUnit.{i}.Version	String(32)
	softwareURL	M	NP	M	String restricted to 1 024	SoftwareComponent/<x>/ Download/<x>/PkgURL (when Download or downloadAndInstall)	chr	Device.SoftwareModules.DeploymentUnit.{i}.URL	String(1024)

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
	swActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the swActionStatus is set to 0 % as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100% and the final state becomes valid.(download_failed or download_complete when the action is "download", install_failed or install_complete when the action is "downloadAndInstall" or "install", remove_failed or remove_complete when the action is "remove", activate_failed or activate_complete when the action is "activate", deactivate_failed or deactivate_complete when the action is "deactivate")		No direct mapping with a BBF element. The progress indicator of the swActionStatus is set to 0% as soon as the action is taken into accountby NREM.. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(install_failed or install_complete when the action is "install", remove_failed or remove_complete when the action is "remove", activate_failed or activate_complete when the action is "activate", deactivate_failed or deactivate_complete when the action is "deactivate")	
SwInstance /softwareAction	swDownload	NP	NP	O	AnyURI	SoftwareComponent/<x>/Download/<x>/Operations/Download	null	N/A. A simple download is not allowed for a TR069 device. An "unauthorized operation" will be returned to a NA that would request a download.	N/A
	swDownloadAnd Install	NP	NP	O	AnyURI	SoftwareComponent/<x>/Download/<x>/Operations/DownloadInstall	null	N/A. A downloadAndInstall is not allowed for a TR069 device. An "unauthorized operation" will be returned to a NA that would request a downloadAndInstall.	N/A
	swInstall	NP	NP	O	AnyURI	SoftwareComponent/<x>/Inventory/Delivered/<x>/Operations/Install	null	Use of an "InstallOpStruct" type or "UpdateOpStruct" type for the "Operations" parameter used in the ChangeDUState RPC	
	swRemove	NP	NP	O	AnyURI	SoftwareComponent/<x>/Inventory/Delivered/<x>/Operations/Remove	null	Use of an "UninstallOpStruct" type for the "Operations" parameter used in the ChangeDUState RPC	
	swActivate	NP	NP	O	AnyURI	SoftwareComponent/<x>/Inventory/Deployed/<x>/Operations/Activate	null	Allows to "start" an EU. NREM use a SetParameterValues RPC with Device.SoftwareModules.ExecutionUnit.{i}.RequestedState = "Active".	

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
	swDeactivate	NP	NP	O	AnyURI	SoftwareComponent/<x>/Inventory/Deployed/<x>/Operations/Deactivate	null	Allows to "stop" an EU. NREM use a SetParameterValues RPC with Device.SoftwareModules.ExecutionUnit.{i}.RequestedState = "Idle".	

E.3.11 Resource etsiReboot

Resource etsiReboot shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.20.

In the BBF TR-069 case, depending on the action, the Reboot or FactoryReset RPC methods are used.

In the OMA case, the reboot or factoryReset operations are performed by using an Exec OMA-DM command.

Table E.20: etsiReboot resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in createReq	M/O in updateReq	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	rebootLevel	O	O	O	Long	DiagMonFcts/Restart/<x>/DiagMonConfig/ConfigParms/RestartLevel	int	N/A	N/A
	rebootTiming	O	O	O	Long	DiagMonFcts/Restart/<x>/DiagMonConfig/ConfigParms/RebootTiming	int	In TR069 the reboot is immediately performed when received by the Device. So, if the NA requests a specific timing, NREM is in charge to start a timer before sending the reboot RPC to the device	
	applicationRef	O	O	O	String	DiagMonFcts/Restart/<x>/DiagMonConfig/ConfigParms/AppReference	chr	N/A	N/A
	rebootActionStatus	NP	NP	M	ActionStatus	No direct mapping with an OMA-DM node. The progress indicator of the swActionStatus is set to 0 % as soon as the action is taken into account by NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(reboot_successful, reboot_failed, reboot_rejected)		No direct mapping with a BBF element. The progress indicator of the rebootActionStatus is set to 0 % as soon as the action is taken into accountby NREM. After completion of the action, the progress indicator is set to 100 % and the final state becomes valid.(reboot_successful, reboot_failed, reboot_rejected)	
rebootAction	reboot	NP	NP	O	AnyURI	DiagMonFcts/Restart/<x>/Operations/Start	null	Use of the "Reboot" RPC.	
	factoryReset	NP	NP	O	AnyURI	LockAndWipe/<x>/Operations/FactoryReset	null	Use of the "FactoryReset" RPC. It has to be noted that this method is optionally supported in a TR069 device.	

E.3.12 Resource etsiAreaNwkInfo

Resource etsiAreaNwkInfo shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.21.

Table E.21: etsiAreaNwkInfo resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in create Req	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
N/A	numOfAreaNwks	NP	NP	M	Long	M2MAreaNwkInfo/NumOfAreaNwks	int	Device.ETSIM2M.SCL.{i}.AreaNwkInstanceNumberOfEntries	unsignedInt
<areaNwkInstance>	areaNwkType	M	NP	M	AreaNwkType	M2MAreaNwkInfo/AreaNwks/<x>/AreaNwkType	chr	Device.ETSIM2M.SCL.{i}.AreaNwkInstance.{i}.AreaNwkType	string
	addressType	M	M	M	String	M2MAreaNwkInfo/AreaNwks/<x>/AddressType	chr	Device.ETSIM2M.SCL.{i}.AreaNwkInstance.{i}.Property.{1}.Value	string
	areaNwkTypeInfo/<areaNwkTypeItem>/name	O	O	O	String	M2MAreaNwkInfo/AreaNwks/<x>/AreaNwkTypeInfo/<x>/ParameterName	chr	Device.ETSIM2M.SCL.{i}.AreaNwkInstance.{i}.Property.{i}.Name	string
	areaNwkTypeInfo/<areaNwkTypeItem>/value	O	O	O	String	M2MAreaNwkInfo/AreaNwks/<x>/AreaNwkTypeInfo/<x>/ParameterValue	chr	Device.ETSIM2M.SCL.{i}.AreaNwkInstance.{i}.Property.{i}.Value	string
<areaNwkInstance>/listOfDevices	members	M	M	M	AnyURI List	M2MAreaNwkInfo/AreaNwks/<x>/ListOfDevices	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.AreaNwkInstance.{i}.ListOfDevices	string (comma separated list of strings)
NOTE: The term N/A in this table refers to the current state of analysis.									

E.3.13 Resource etsiAreaNwkDeviceInfo

Resource etsiAreaNwkDeviceInfo shall be mapped to OMA DM and BBF TR-069 objects as specified in table E.22.

Table E.22: etsiAreaNwkDeviceInfo resource tree mapping to OMA DM and BBF TR-069

Parameter set (TC M2M <parameters> subresource)	Attributes (TC M2M)	M/O in createReq	M/O in update Req	M/O in resp	Type (TC M2M)	Node (OMA DM)	Type (OMA DM)	Element (BBF TR-069)	Type (BBF)
<areaNwkDeviceInfoInstance>	areaNwkID	M	NP	M	Long	<x>/AreaNwks/<x>/AreaNwkID	chr	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.AreaNwkInstance	string
	sleepInterval	O	O	O	Long	<x>/AreaNwks/<x>/SleepInterval	int	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.SleepInterval	unsigned Int
	sleepDuration	O	O	O	Long	<x>/AreaNwks/<x>/SleepDuration	int	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.SleepDuration	unsigned Int
	status	NP	NP	O	AreaNwkStatus	<x>/AreaNwks/<x>/Status	chr	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.Status	string
	areaNwkTypeInfoOfDevice/<areaNwkTypeItem>/name	O	O	O	String	<x>/AreaNwks/<x>/AreaNwkTypeInfoOfDevice/<x>/ParameterName	chr	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.Property.{i}.Name	string
	areaNwkTypeInfoOfDevice/areaNwkTypeItem/value	O	O	O	String	<x>/AreaNwks/<x>/AreaNwkTypeInfoOfDevice/<x>/ParameterValue	chr	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.Property.{i}.Value	string
<areaNwkDeviceInfoInstance> /groups/ listOfDeviceNeighbors	members	O	O	O	AnyURIList	<x>/AreaNwks/<x>/Groups/ListOfDeviceNeighbors	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.ListOfDeviceNeighbors	string (comma separated list of strings)
<areaNwkDeviceInfoInstance> /groups/ listOfDeviceApplications	members	O	O	O	AnyURIList	<x>/AreaNwks/<x>/Groups/ListOfDeviceApplications	chr (comma separated list of strings)	Device.ETSIM2M.SCL.{i}.AreaNwkDeviceInfoInstance.{i}.ListOfDeviceApplications	string (comma separated list of strings)
NOTE: The term N/A in this table refers to the current state of analysis.									

Annex F (normative): Interworking with XDMS

This annex is normative if and only if an XDMS is deployed for the purpose of managing the M2M resources.

F.1 Application Usage of XDMS for Management of M2M Service Capabilities Resources

F.1.1 High level Architectural Principles

An M2M service provider shall apply the directory scheme shown below. In this scheme, the following can be noted:

- There are multiple sclBases under the XCAP root.
- All sclBases shall share the same Application usages (e.g. AUIDm, AUID1 in figureF.1).
- OMA defined Application usages shall retain their location in the M2M directory scheme. In other words, xcap-caps AUID shall be located immediately beneath the XCAP root. The same applies to the oma.openmpobilealliance.xcap-directory AUID, which shall be located immediately beneath the XCAP root.

Figure F.1 shows a high level overview of the above principles.

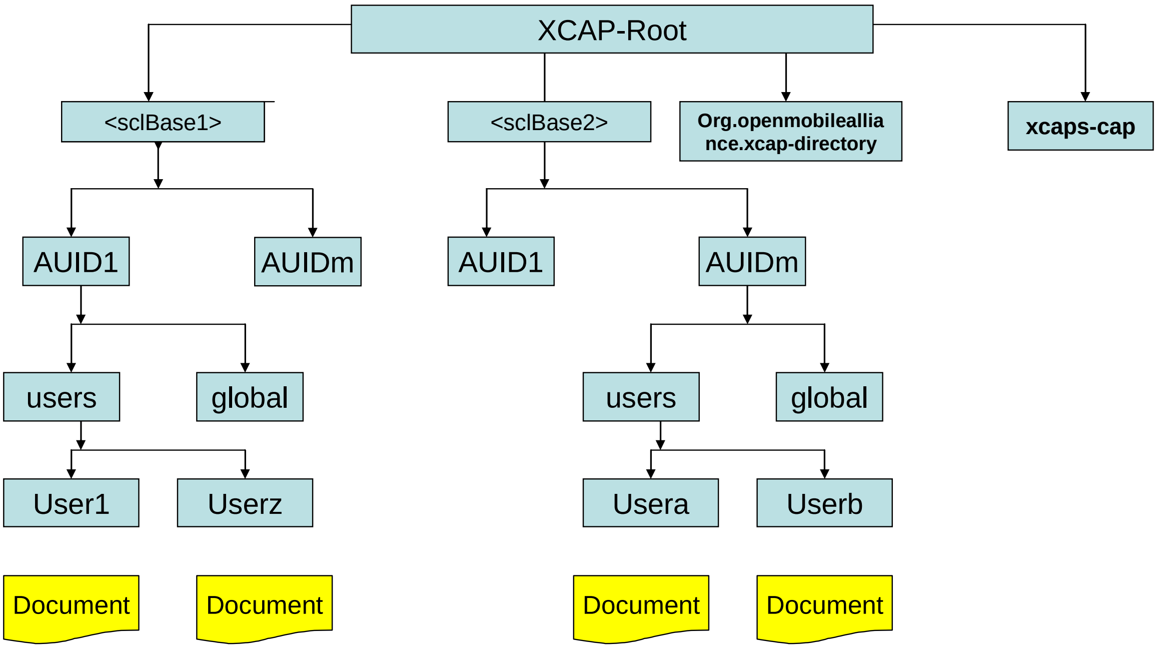


Figure F.1: XDMS Framework for M2M Resources

F.1.1.1 Mapping Principles

The following additional principles apply in the various application usages:

- Each M2M URI representing a collection shall be represented as an empty XDM document in XDMS. The XDM document name is returned as a reference representing the collection. In that respect, it is acting as a dummy resource.
- M2M URI of children (sub-resources) in a collection resource is stored as XCAP references, each in their own document based on collection type. Every collection has an application usage defined for that purpose.

- M2M URI of children (sub-resources) that are not a collection includes a reference which can either be an XCAP reference to some XDM resource, or an M2M URI reference (e.g. latest, oldest). depending on the application usage.
- M2M URIs in a resource that refers to non-children (attribute) is mapped to a separate document entitled Index document. There are some exceptions; they are specified in the applicable application usages.
- AccessRightID is mapped in a special document entitled accessPermission that includes an XCAP reference to an access right resource defined by the Access Right Application Usage. The handling of access rights is more described in clause F.2.1.4.
- When creating collections, where applicable, the targetID shall be used for creating those collections in XDMS. This implies that the XUI for the parent and the collection is identical. The targetID for the system startup procedure in clause F.2.1.25 shall be /sclBase/.
- When a parent resource is created, all its sub-resources as well as non-sub resource elements are created at the same time (see the various procedures in clause F.2.1).

F.1.2 M2M Service Capabilities Application Usages

Figures F.2 to F.5 show the proposed directory structure for the M2M resources.

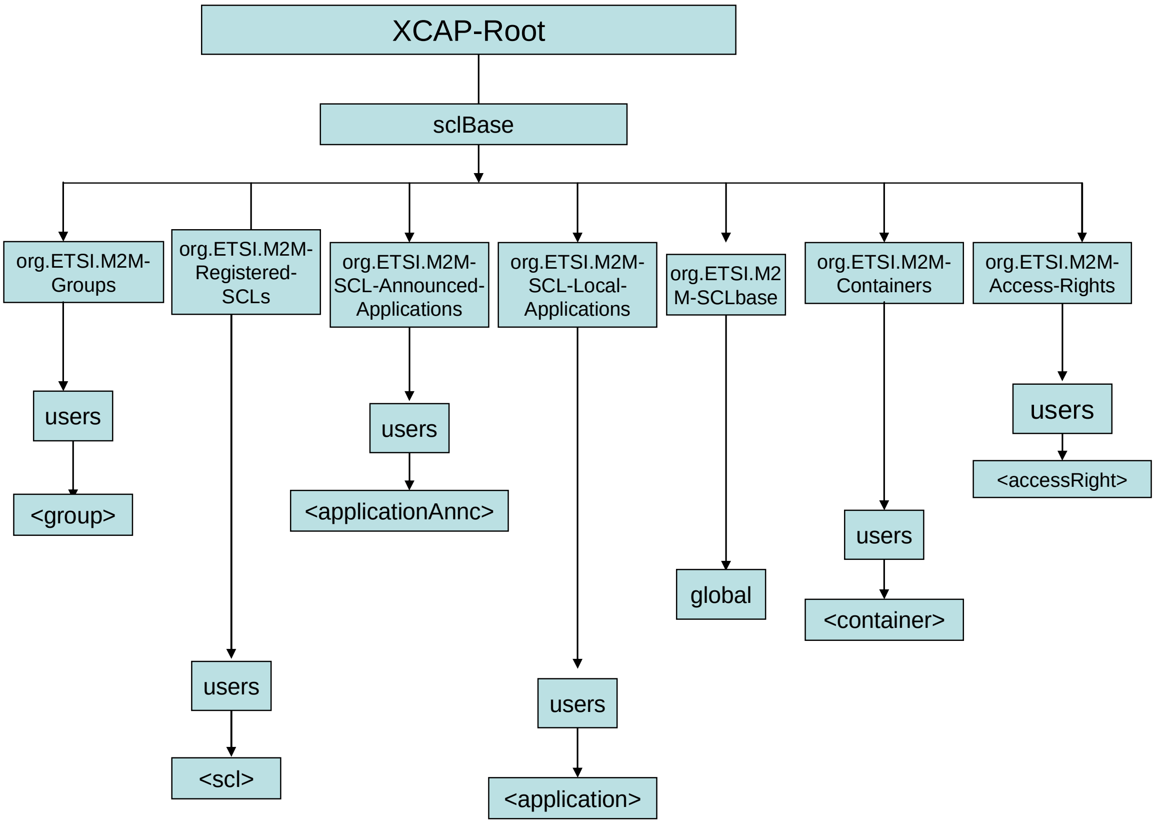


Figure F.2: Part 1 - Overall Directory Structure

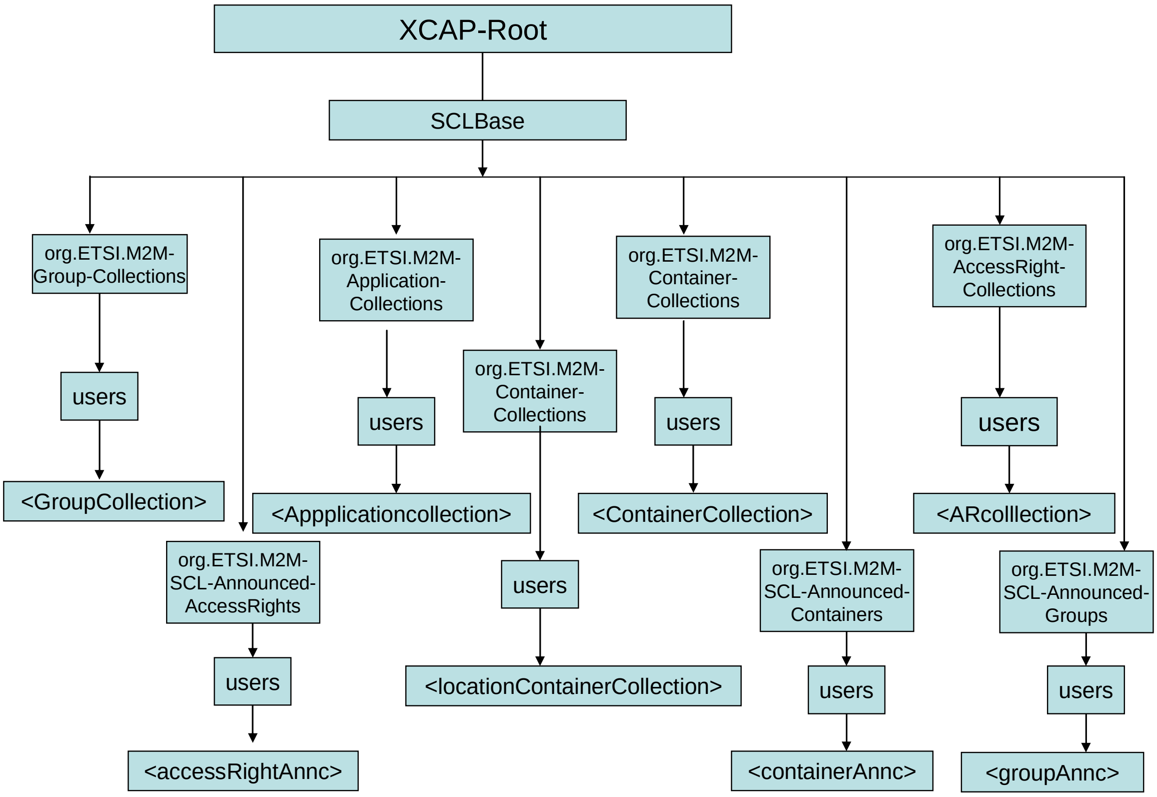

XCAP-Root
SCLBase
org.ETSI.M2M-Group-Collections
users
<GroupCollection>
org.ETSI.M2M-Application-Collections
users
<Appplicationcollection>
org.ETSI.M2M-Container-Collections
users
<locationContainerCollection>
org.ETSI.M2M-Container-Collections
users
<ContainerCollection>
org.ETSI.M2M-AccessRight-Collections
users
<ARcolllection>
org.ETSI.M2M-SCL-Announced-AccessRights
users
<accessRightAnnc>
org.ETSI.M2M-SCL-Announced-Containers
users
<containerAnnc>
org.ETSI.M2M-SCL-Announced-Groups
users
<groupAnnc>

Figure F.3: Part 2 - Overall Directory Structure

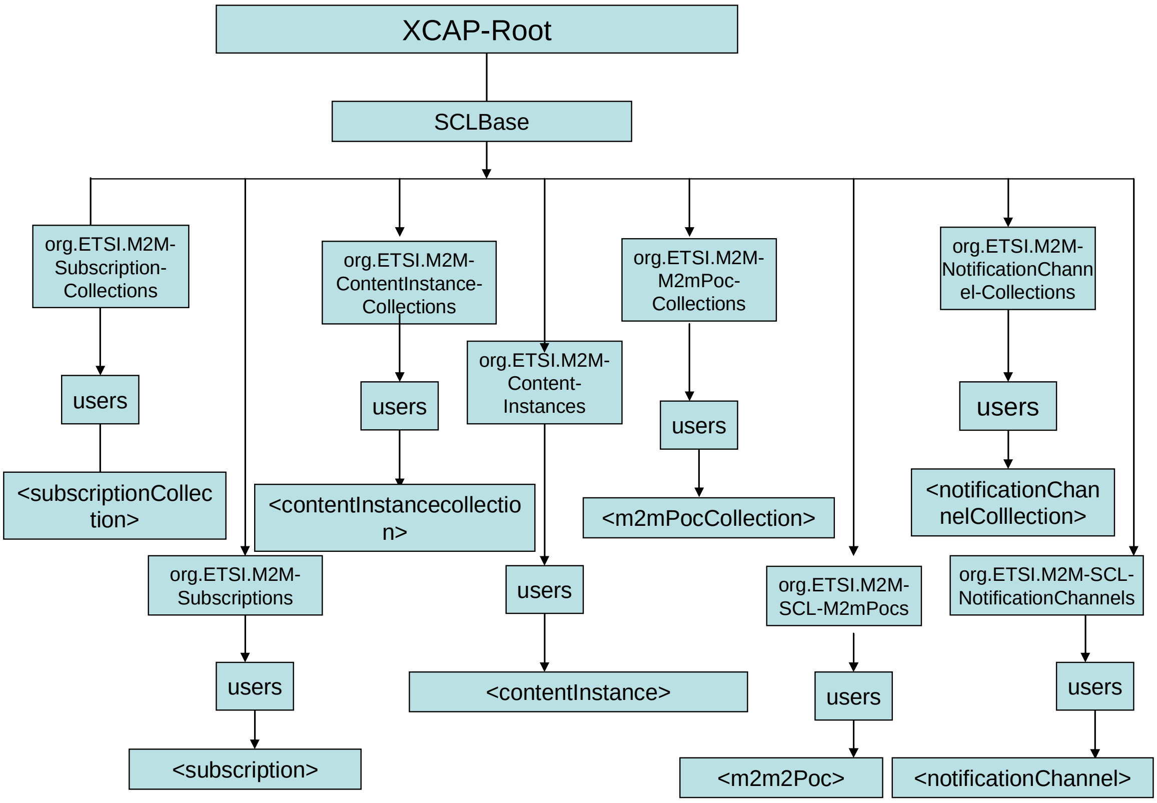

XCAP-Root
SCLBase
org.ETSI.M2M-Subscription-Collections
users
<subscriptionCollection>
org.ETSI.M2M-ContentInstance-Collections
users
<contentInstancecollection>
org.ETSI.M2M-Content-Instances
users
<contentInstance>
org.ETSI.M2M-M2mPoc-Collections
users
<m2mPocCollection>
org.ETSI.M2M-NotificationChannel-Collections
users
<notificationChannelColllection>
org.ETSI.M2M-Subscriptions
users
<subscription>
org.ETSI.M2M-SCL-M2mPocs
users
<m2m2Poc>
org.ETSI.M2M-SCL-NotificationChannels
users
<notificationChannel>

Figure F.4: Part 3 - Overall Directory Structure

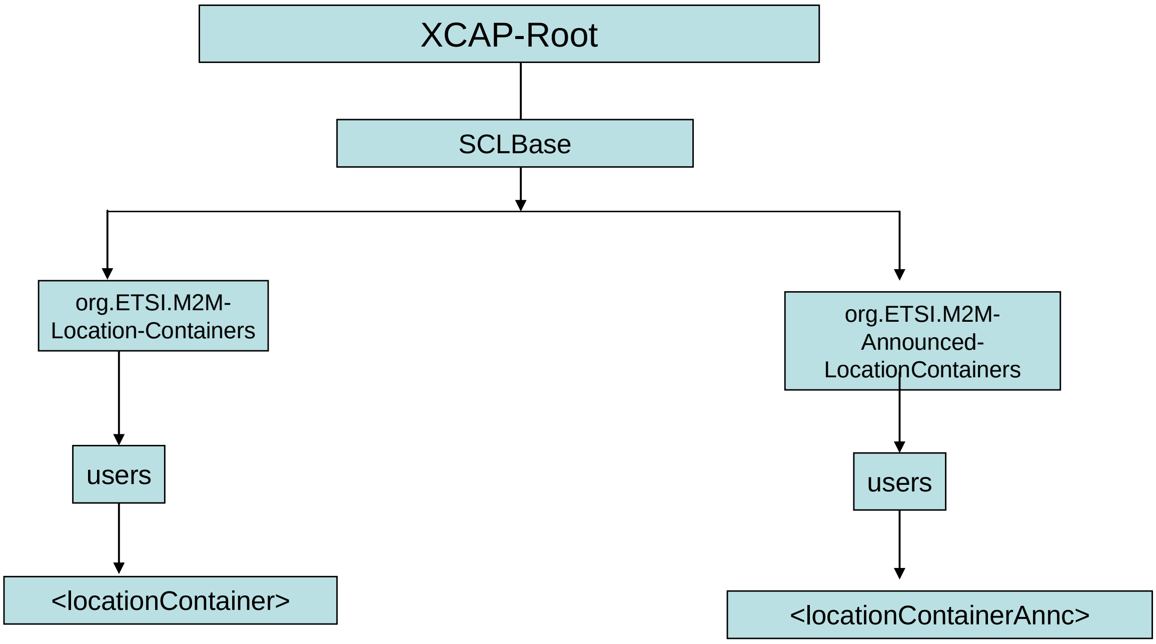

XCAP-Root
SCLBase
org.ETSI.M2M-Location-Containers
users
<locationContainer>
org.ETSI.M2M-Announced-LocationContainers
users
<locationContainerAnnc>

Figure F.5: Part 4 - Overall Directory Structure

Twenty Five Application Usages are identified for managing M2M resources. They are as follows:

- Access Right Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M access rights resources created under the sclBase.

- Group Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M group resources created under the sclBase tree.

- Container Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M container resources created under the sclBase tree.

- The Registered SCL (remote gateways/devices) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M SLCs (registered SCLs) created under the sclBase tree.

- Registered M2M Applications Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M applications created under the sclBase tree.

- Announced Applications (Remotely registered M2M Applications) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M announced applications resources created under the sclBase tree.

- Container Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M container collection resources.

- Group Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M group collection resources.

- Access Rights Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M access rights collection resources.

- Application Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M application collection resources.

- M2M SCLbase Application usage defined under the sclBase tree. This Application Usage handles the management of sclBase resources.

- SCL Collection Application usage defined under the sclBase tree. This Application Usage handles the management of all M2M scl collection resources created under the sclBase tree.

- Location Container Application usage defined under the sclBase tree. This Application Usage handles the management of M2M location container resources created under the sclBase tree.

- Announced Groups (remotely created groups) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M announced groups created under the sclBase tree.

- Announced Containers (remotely created containers) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M announced containers created under the sclBase tree.

- Announced Access Rights (remotely created accessrights) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M announced access rights created under the sclBase tree.

- Announced locationContainers (remotely created location containers) Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M announced location containers created under the sclBase tree.

- Subscription Collection Application usage defined under the sclBase tree. This Application Usage handles the management of all M2M subscription collection resources created under the sclBase tree.

- m2mPoc Collection Application usage defined under the sclBase tree. This Application Usage handles the management of all M2M m2mPoc collection resources created under the sclBase tree.

- NotificationChannel Collection Application usage defined under the sclBase tree. This Application Usage handles the management of all M2M notificationChannel collection resources created under the sclBase tree.

- ContentInstance Collection Application usage defined under the sclBase tree. This Application Usage handles the management of all M2M contentInstance collection resources created under the sclBase tree.

- Subscription Application usage defined under the sclBase tree. This Application Usage handles the management of M2M subscription resources created under the sclBase tree.

- M2mPoc Application usage defined under the sclBase tree. This Application Usage handles the management of M2M m2mPoc resources created under the sclBase tree.

- ContentInstance Application usage defined under the sclBase tree. This Application Usage handles the management of M2M contentInstance resources created under the sclBase tree.

- NotificationChannel Application usage defined under the sclBase tree. This Application Usage handles the management of M2M notificationChannel resources created under the sclBase tree.

Note that in all Application Usages dealing with Collections, none of the referenced resources included in any collection are referenced more than once. It is the responsibility of the NSCL to enforce this restriction.

F.1.2.1 SCLBase Application Usage

The SCLBase application usage is an application usage that includes all resources defined under the <sclBase> as specified in TS 102 690 [2] with the exception of scls resources.

This application usage allows an M2M entity that have the appropriate access rights to create/modify/delete M2M resources stored under this application usage.

This application usage defines only one global tree.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-SCLBase".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-SCLBase".

XML Schema

The SCLBase XDM documents shall conform to the following XML schema:

- Attributes document under global tree shall conform to the XSD defined in the present document, annex B, as specified in sclBase.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Containers document under global tree shall be an empty document.
- Groups document under global tree shall be an empty document.
- Scls document under global tree shall be an empty document.
- Subscriptions document under global tree shall be an empty document.
- AccessPermission document under global tree shall conform to an XCAP reference as depicted in table F.2.
- Application document under global tree shall be an empty document.
- AccessRights document under global tree shall be an empty document.

Additional Constraints

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for every requested operation on any resource under the <SCLBase> tree before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall not be any instance of the present document in the user's tree.

Authorization Policies

The XDMS server shall authorize requests for all operations on all XDM documents located under the entire SCLBase global tree using the accessPermission document located under the global tree for that purpose against the request originator included in the X-3GPP-Asserted-Identity HTTP header.

Global Document

This Application Usage defines eight global documents. The first XDM document includes the attributes for the SCLBase. The well-known name of the document is attributes. The attributes document shall be addressed using the global directory document selector "/Attributes/, i.e. the document selector to the XDM attributes document shall be "[auid]/global/Attributes/attributes".

The second XDM document represents the SCLBase collection of containers that don't have a containment relation with any specific entity. The well-known name of the document is containers. The containers document shall be addressed using the global directory document selector "/Containers/, i.e. the document selector to the XDM containers document shall be "[auid]/global/Containers/containers".

The third XDM document represents the SCLBase collection of groups that do not have a containment relation with any specific entity. The well-known name of the document is groups. The groups document shall be addressed using the global directory document selector "/Groups/, i.e. the document selector to the XDM groups document shall be "[auid]/global/Groups/groups".

The fourth XDM document represents the SCLBase access permission document which governs access rights for users to XDM documents under the global tree. The well-known name of the document is accessPermission. The accessPermission document shall be addressed using the global directory document selector "/Access-Permission/, i.e. the document selector to the XDM accessPermission document shall be "[auid]/global/Access-Permission/accessPermission".

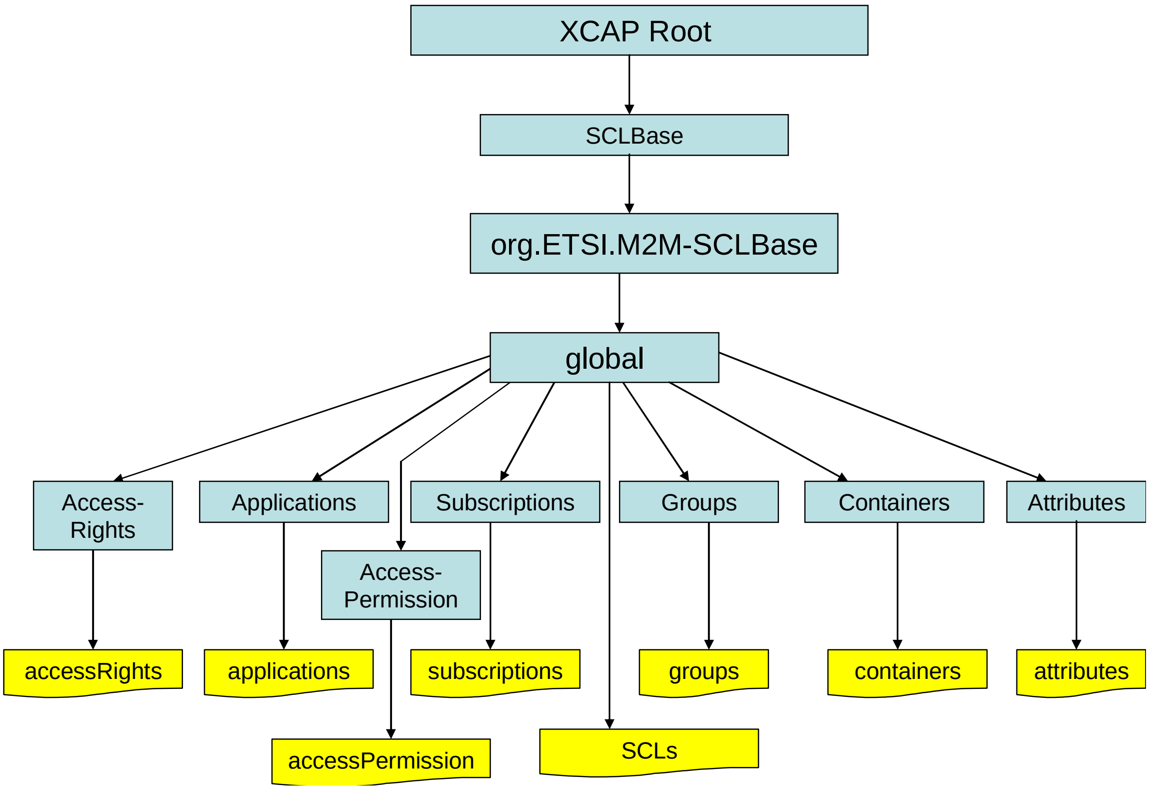
The fifth XDM document represents the SCLBase collection of subscriptions to XDM resources under the global tree. The well-known name of the document is subscriptions. The subscriptions document shall be addressed using the global directory document selector "/Subscriptions/, i.e. the document selector to the XDM subscriptions document shall be "[auid]/global/Subscriptions/subscriptions".

The six XDM document represents the SCLBase collection of applications registered under the SCLBase. The well-known name of the document is applications. The applications document shall be addressed using the global directory document selector "/Applications/, i.e. the document selector to the XDM applications document shall be "[auid]/global/Applications/applications.

The seventh XDM document represents the SCLBase collection of access-rights that don't have a containment relation with any specific entity. The accessRights document shall be addressed using the global directory document selector "/Access-Rights/, i.e. the document selector to the XDM accessRights document shall be "[auid]/global/Access-Rights/accessRights".

The last XDM document represents the SCLBase collection of SCLs registered under the SCLBase. The well-known name of the document is scls. The scls document shall be addressed using the global directory document selector "/SCLs/, i.e. the document selector to the XDM scls document shall be "[auid]/global/SCLs/scls".

There is only a single instance for all these documents.



NOTE: Discovery resource need to be saved here.

Figure F.6: Tree Structure for SCLBase Application Usage

F.1.2.1.1 Mapping between the XDMS XCAP resources and M2M sclBase resources

Table F.1 depicts the mapping between XDMS XCAP resources and M2M sclBase resource tree as defined in TS 102 690 [2]. The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-SCLBase/global/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.1: sclBase M2M Resource URL Mapping <-> XCAP URL

M2M Resource URL	XDMS XCAP URL
<sclBase>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/Attributes/attributes
<sclBase>/containers	XCAPbase/Containers/containers
<sclBase>/groups	XCAPbase /Groups/groups
<sclBase>/subscriptions	XCAPbase/Subscriptions/subscriptions
<sclBase>/accessRights	XCAPbase/Access-Rights/accessRights
<sclBase>/applications	XCAPbase/Applications/applications
<sclBase>/scls	XCAPbase/SCLs/scls
<sclBase>/attribute (AccessRightID)	XCAPbase/accessPermission

F.1.2.1.2 Information Mapping between XDMS XCAP resources and M2M Resources

Table F.2 depicts the mapping between XML data stored in XDMS XCAP resources and XML data for M2M resources as defined in the present document, annex B.

Table F.2: sclBase XDMS Resource <-> M2M Resource

XDMS XCAP Resource	M2M Resource
XCAPbase/Attributes/attributes	XDM document conforms to the XML schema for all attributes defined under <sclBase>attributes excluding accessRightID attribute
XCAPbase/Containers/containers	Empty Document
XCAPbase/Groups/groups	Empty Document
XCAPbase/Subscriptions/subscriptions	Empty Document
XCAPbase/Access-Rights/accessRights	Empty Document
XCAPbase/Applications/applications	Empty Document
XCAPbase/SCLs/scls	Empty Document
XCAPbase/Access-Permission/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.2 Management of Registered SCL resources

F.1.2.2.1 Registered SCL Application Usage

The Registered SCL Application Usage allows an M2M entity to be able to create/modify/delete a Remote SCL resource registered locally.

Every Registered SCL resource that is created shall be allocated a unique identity and shall be located below the users' tree located beneath the AUID tree allocated to Registered SCL resources.

Application Unique ID

The AUID SHALL be "org..ETSI.M2M-Registered-SCLs".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Regsitered-SCLs".

XML Schema

The Registered SCL XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in scls.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document ,per XUI, under users' tree shall conform to an XCAP reference as depicted in table F.4.
- Containers document ,per XUI, under users' tree shall be an empty document.
- Groups document ,per XUI, under users' tree shall be an empty document.
- Applications document ,per XUI, under users' tree shall be an empty document.
- AccessRights document ,per XUI, under users' tree shall be an empty document.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as depicted in table F.4.
- NotificationChannels document, per XUI, under users' tree shall be an empty document.
- M2MPocs document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a registered SCL resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be nine well-known documents. They are as follows:

- The well-known name of the main Registered SCL resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "containers ". The document selector to access the "containers" XDM document shall be "[auid]/users/[xui]/containers".
- The fourth well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fifth well-known document shall be "groups". The document selector to access the "groups" XDM document shall be "[auid]/users/[xui]/groups".
- The sixth well-known document shall be "accessRights". The document selector to access the "accessRights" XDM document shall be "[auid]/users/[xui]/accessRights".

- The seventh well-known document shall be "applications". The document selector to access the "applications" XDM document shall be "[auid]/users/[xui]/applications".
- The eights well-known document shall be "notificationChannels". The document selector to access the :notificationChannels" XDM document shall be "[auid]/users/[xui]/notificationChannels".
- Finally, the last well-known document shall be "m2mPocs". The document selector to access the m2mPocs XDM document shall be "[auid]/users/[xui]/m2mPocs".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents for this application usage.

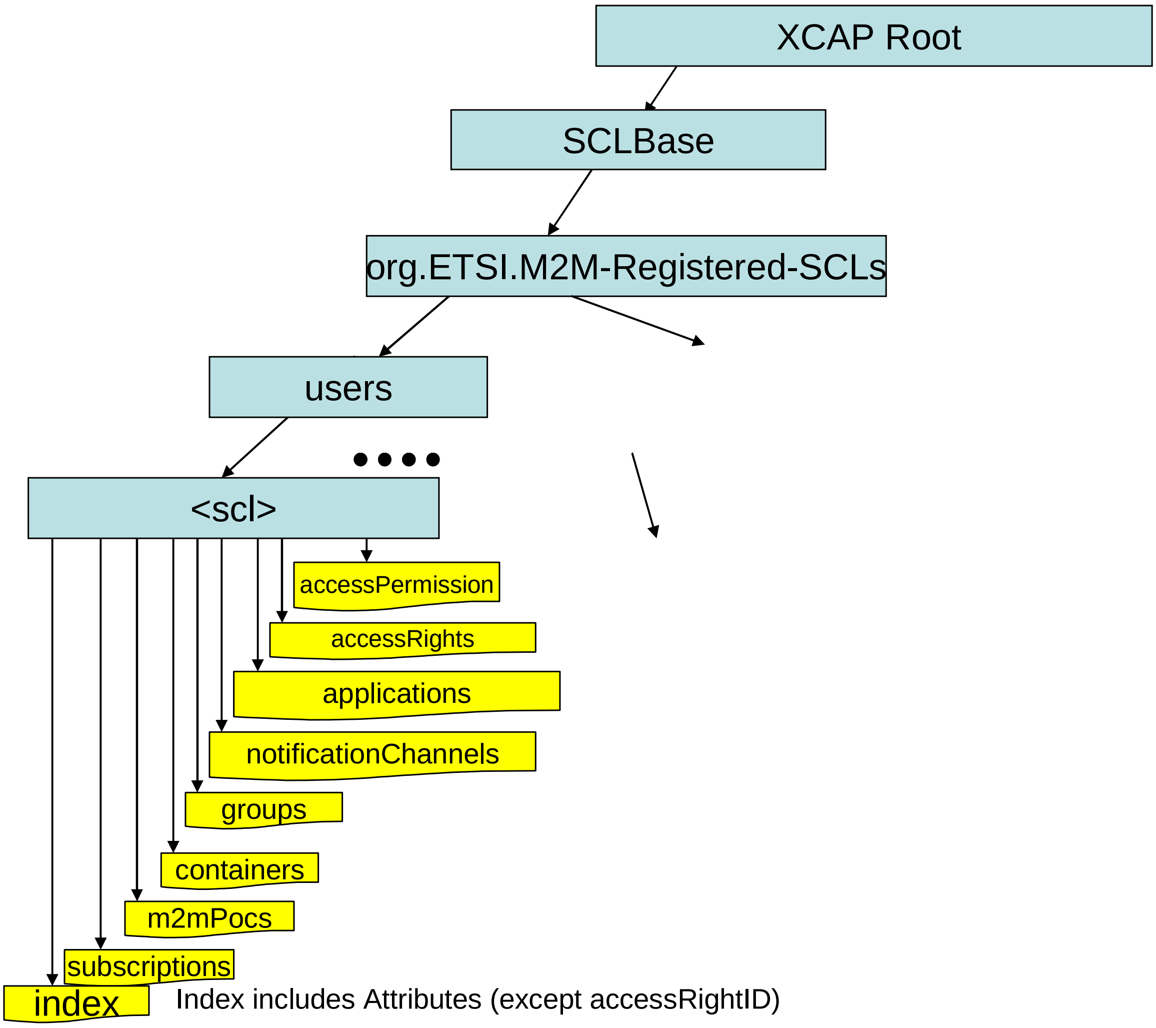


Figure F.7: Tree Structure for Registered SCLs

F.1.2.2.2 Mapping between XDMS XCAP <scl> resources and M2M <scl> resources

It is to be noted that a 1:1 mapping exists between M2M <scl> resources and XDMS XCAP <scl> resources. The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.3: Registered SCL M2M Resource URL Mapping <-> XCAP URL

M2M <scl> Resources	XDMS XCAP <scl> Resources
<sclBase>/scls/<scl>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/containers	XCAPbase/<XUI>/containers
<sclBase>/scls/<scl>/groups	XCAPbase/<XUI>/groups
<sclBase>/scls/<scl>/subscriptions	XCAPbase/<XUI>/subscriptions
<sclBase>scls/<scl>/accessRights	XCAPbase/<XUI>/accessRights
<sclBase>scls/<scl>/applications	XCAPbase/<XUI>/applications
<sclBase>scls/<scl>/m2mPocs	XCAPbase/<XUI>/m2mPocs
<sclBase>scls/<scl>/notificationChannels	XCAPbase/<XUI>/notificationChannels
<sclBase>/scls<scl>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.2.3 Information Mapping between the XDMS XCAP <scl> resources and M2M <scl> resources

Table F.4: Registered SCL XDMS Resource <-> M2M Resource

XDMS XCAP <scl> Resource	M2M <scl> Resource
XCAPbase/<XUI>/index	XDM document conforms to XML schema for all attributes defined for the <scl> resource except accessRightID
XCAPbase/<XUI>/containers	Empty Document
XCAPbase/<XUI>/groups	Empty Document
XCAPbase/<XUI>/accessRights	Empty Document
XCAPbase/<XUI>/m2mPocs	Empty Document
XCAPbase/<XUI>/notificationChannels	Empty Document
XCAPbase/<XUI>/subscriptions	Empty Document
XCAPbase/<XUI>/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Right/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here
XCAPbase/<XUI>/applications	Empty Document

F1.2.3 Management of Announced Application resources

F.1.2.3.1 Announced Applications Application Usage

The Announced Applications Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an Announced Applications resources.

Every Announced Applications resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the Announced Applications resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Announced-Applications".

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Announced-Applications".

XML Schema

The Announced Applications XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in applicationAnnc.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.

- Containers document, per XUI, under users' tree shall be an empty document.
- Groups document, per XUI, under users' tree shall be an empty document.
- AccessRights document, per XUI, under users' tree shall be an empty document.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.6.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an M2M Announced Applications resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be five well-known documents. They are as follows:

- The well-known name of the main Announced Application resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "groups". The document selector to access the "groups" XDM document shall be "[auid]/users/[xui]/groups".
- The third well know document shall be "containers". The document selector to access the "containers" XDM document shall be "[auid]/users/[xui]/containers".
- The fourth well-known document shall be "accessRights". The document selector to access the "accessRights" XDM document shall be "[auid]/users/[xui]/accessRights".
- Finally, the last well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

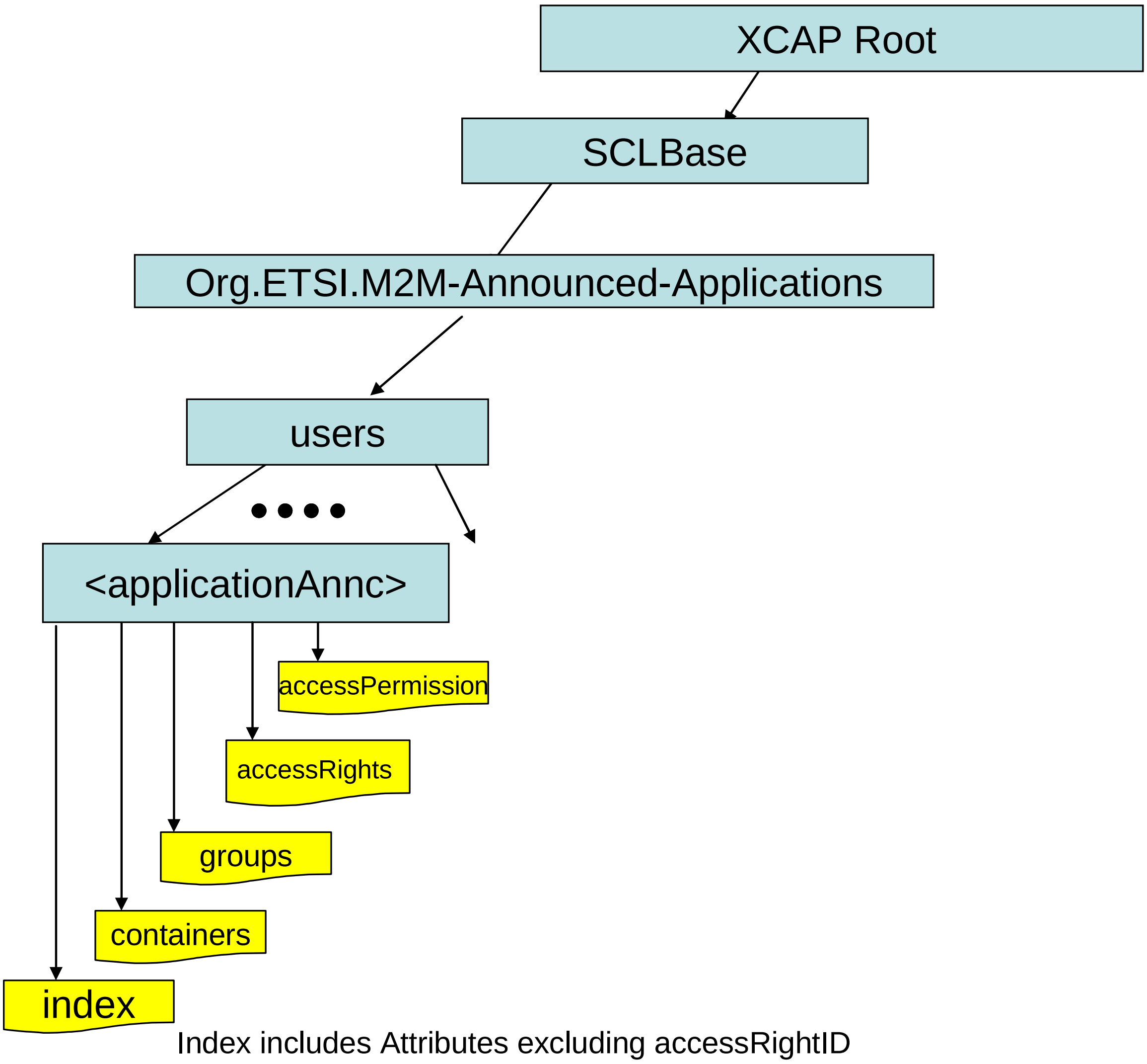


Figure F.8: Tree Structure for Application Announcement Application Usage

F.1.2.3.2 Mapping between the XDMS XCAP <applicationAnnc> resources and M2M <applicationAnnc> resources

It is to be noted that a 1:1 mapping exists between M2M <applicationAnnc> resources and XDMS XCAP <applicationAnnc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.5: Announced Application M2M Resource URL Mapping <-> XCAP URL

M2M <applicationAnnc> Resources	XDMS XCAP <applicationAnnc> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers	XCAPbase/XUI/containers
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups	XCAPbase/XUI/groups
<sclBase>scls/<scl>/applications/<applicationAnnc>/accessRights	XCAPbase/XUI/accessRights
<sclBase>/scls<scl>/applications/<applicationAnnc>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.3.3 Information Mapping Between XDMS XCAP <applicationAnnc> resources and M2M <applicationAnnc> resources

Table F.6: Announced Application XDMS Resource <-> M2M Resource

XDMS XCAP <applicationAnnc> Resource	M2M <applicationAnnc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <applicationAnnc> resource except accessRightID
XCAPbase/XUI/containers	Empty Document
XCAPbase/XUI/groups	Empty Document
XCAPbase/XUI/accessRights	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.4 Management of Local Application resources

F.1.2.4.1 Local Applications Application Usage

The Local Applications Application Usage allows an M2M entity to be able to Create/Modify /Delete/Read an M2M Local Application resource.

Every M2M Local Application resource that is created shall be allocated a unique identity and shall be located under the users' tree beneath the AUID tree allocated to the Local Applications resources.

Applications managed under this resource are applications that are registered locally.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Local-Applications".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Local-Applications".

XML Schema

The Local Applications XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in application.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document under users' tree, per XUI, shall be an empty document.
- Containers document under users' tree, per XUI, shall be an empty document.

- Groups document under users' tree, per XUI, shall be an empty document.
- AccessRights document under users' tree, per XUI, shall be an empty document.
- AccessPermission document under users' tree, per XUI, shall conform to an XCAP reference as described in table F.8.
- NotificationChannels document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Local Application resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be seven well-known documents. They are as follows:

- The well-known name of the main Local Application resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "containers". The document selector to access the "containers" XDM document shall be "[auid]/users/[xui]/containers".
- The fourth well-known document shall be "groups". The document selector to access the "groups" XDM document shall be "[auid]/users/[xui]/groups".
- The fifth well-known document shall be "accessRights". The document selector to access the "accessRights" XDM document shall be "[auid]/users/[xui]/accessRights".
- The sixth well-known document shall be "notificationChannels". The document selector to access the "notificationChannels" XDM document shall be "[auid]/users/[xui]/notificationChannels".
- Finally, the last well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the access-rights document under the same XUI tree for that purpose.

Global Document

There are no global documents for this application usage.

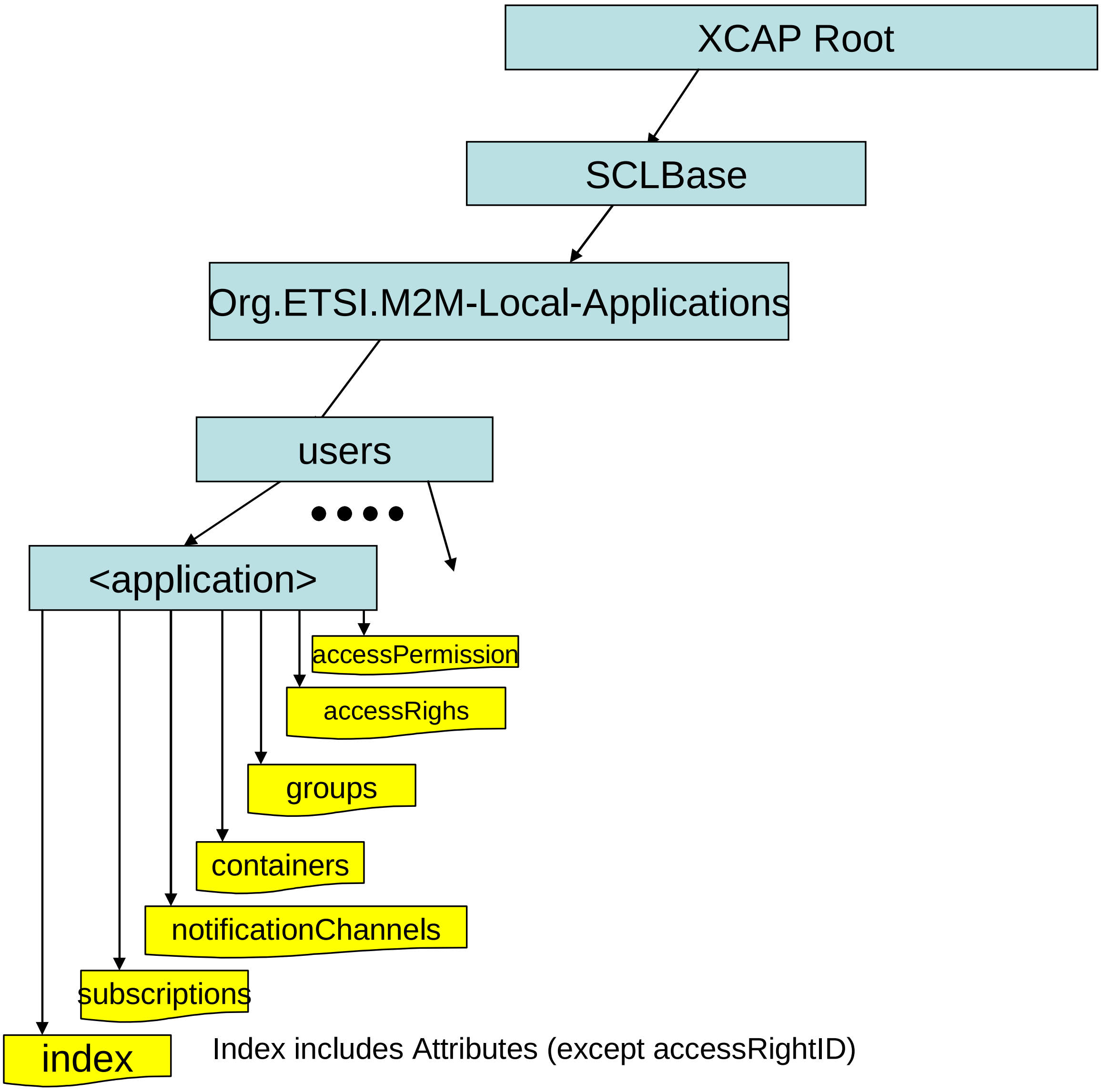


Figure F.9: Tree structure for Local SCL applications Application Usage

F.1.2.4.2 Mapping between the XDMS XCAP <application> resources and M2M <application> resources

There are multiple locations within the M2M Resource tree structure where <application> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <application> and M2M resources. However, only one case shall be described here.

It is to be noted that a 1:1 mapping exists between M2M <application> resources and XDMS XCAP <application> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.7: Local Application M2M Resource URL Mapping <-> XCAP URL

M2M <application> Resource URL	XDMS XCAP <application> Resource URL
<sclBase>/applications/<application>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>/containers	XCAPbase/XUI/containers
<sclBase>/applications/<application>/groups	XCAPbase/XUI/groups
<sclBase>/applications/<application>/accessRights	XCAPbase/XUI/accessRights
<sclBase>/applications/<application>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>applications/<application>/notificationChannels	XCAPbase/<XUI>/notificationChannels
<sclBase>/applications/<application>/attribute (AccessRightID)	XCAPbase/XUI/accessPermission

F.1.2.4.3 Information Mapping Between XDMS XCAP <application> resource and M2M <application> resource

Table F.8: Local Application XDMS Resource <-> M2M Resource

XDMS XCAP <application> Resource	M2M <application> Resource
XCAPbase/XUI/Index	XDM document conforms to the XML schema for all attributes defined for the <application> resource except accessRightID
XCAPbase/XUI/containers	Empty Document
XCAPbase/XUI/groups	Empty Document
XCAPbase/XUI/accessRights	Empty Document
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/<XUI>/notificationChannels	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.5 Management of Access Right Resources

F.1.2.5.1 Access Right Resource Application Usage

The Access Right Application Usage allows an M2M entity to create/modify/delete Access Right resources.

Each Access Right resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Access Rights resources.

The appropriate Access Right Resources shall be referenced, where needed, to authorize any operation on any M2M resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Access-Rights".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Access-Rights".

XML Schema

The Access Right resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in accessRight.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.

- Subscriptions document, per XUI, under user' tree shall be an empty document.
- Permissions document, per XUI, under user' tree shall conform to XML schema as defined in table F.13.
- AccessPermission document, per XUI, under user' tree shall conform to XML schema as defined in table F.13.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an Access Right resource under the users' tree is unique.

For any incoming request regarding an operation on Access Right resources, the request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted. The accessPermission document shall be used for that purpose.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There are four well-known documents defined. They are as follows:

- The first well-known XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well known document shall be "permissions". The document selector to access the "permissions" XDM document shall be "[auid]/users/[xui]/permissions".
- Finally, the last well known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document located under the same XUI tree for that purpose.

Global Document

There are no global documents defined under the global tree.

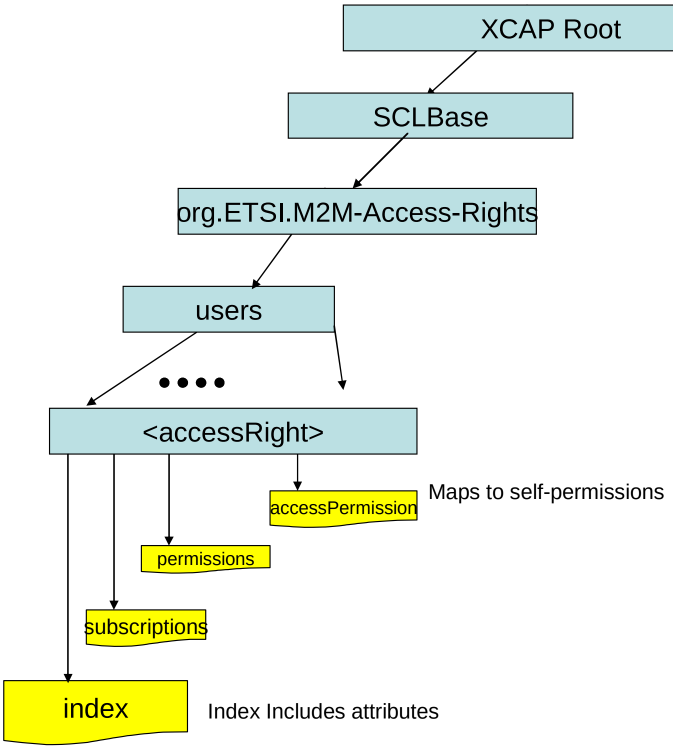


Figure F.10: Tree structure for Access Rights Application Usage

F.1.2.5.2 Mapping between the XDMS XCAP <accessRight> Resources and M2M <accessRight> resources

There are multiple locations within the M2M Resource tree structure where <accessRight> resources are defined. As such there are multiple cases for mapping between XDMS XCAP <access Right> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <accessRight> resources and XDMS XCAP <accessRight> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.9: Access Right M2M Resource URL Mapping <-> XCAP URL

M2M <accessRight> Resource	XDMS XCAP <accessRight> Resource
<sclBase>/scls/<scl>/accessRights/<accessRight>/attribute (Includes all attributes except permissions, and selfPermisisions)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/accessRights/<accessRight>/permissions	XCAPbase/XUI/permissions
<sclBase>/scls/<scl>/accessRights/<accessRight>/selfPermissions	XCAPbase/XUI/accessPermission
<sclBase>/scls/<scl>/accessRights/<accessRight>/subscriptions	XCAPbase/XUI/subscriptions

Case 2:

Table F.10: Access Right M2M Resource URL Mapping <-> XCAP

M2M <accessRight> Resource URL	XDMS XCAP <accessRight> Resource URL
<sclBase>/accessRights/<accessRight>/attribute (Includes all attributes except permissions, and selfPermisisions)	XCAPbase/XUI/index
<sclBase>/accessRights/<accessRight>/permissions	XCAPbase/XUI/permissions
<sclBase>/accessRights/<accessRight>/selfPermissions	XCAPbase/XUI/accessPermission
<sclBase>/accessRights/<accessRight>/subscriptions	XCAPbase/XUI/subscriptions

Case 3:

Table F.11: Access Right M2M Resource URL Mapping <-> XCAP

M2M <accessRight> Resource URL	XDMS XCAP <accessRight> Resource URL
<sclBase>/applications/<application>/accessRights/<accessRight>/attribute (Includes all attributes except permissions, and selfPermissions)	XCAPbase/XUI/index
<sclBase>/applications/<application>/accessRights/<accessRight>/permissions	XCAPbase/XUI/permission
<sclBase>/applications/<application>/accessRights/<accessRight>/selfPermissions	XCAPbase/XUI/accessPermission
<sclBase>/applications/<application>/accessRights/<accessRight>/subscriptions	XCAPbase/XUI/subscriptions

Case 4:

Table F.12: Access Right M2M Resource URL Mapping <-> XCAP

M2M <accessRight> Resource URL	XDMS XCAP XUI Resource URL
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/<accessRight>/attribute (Includes all attributes except permissions, and selfPermissions)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/<accessRight>/permissions	XCAPbase/XUI/permissions
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/<accessRight>/selfPermissions	XCAPbase/XUI/accessPermission
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/<accessRight>/subscriptions	XCAPbase/XUI/subscriptions

F.1.2.5.3 Information Mapping between XDMS XCAP <accessRight> resource and M2M <accessRight> Resource

Table F.13: Access Right XDMS Resource <-> M2M Resource

XDMS XCAP <accessRight> Resource	M2M <accessRight> Resource
XCAPbase/XUI/index (Includes all attributes except permissions, and selfPermissions)	XDM document conforms to the XML schema for all defined attributes
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/permissions	XDM document conforms to the XML schema corresponding to permissions attribute resource
XCAPbase/XUI/accessPermission	XDM document conforms to the XML schema corresponding to selfPermissions attribute resource

F.1.2.6 Management of Group Resources

F.1.2.6.1 Group Application Usage

The Group Application Usage allows an M2M entity to be able to create/modify/delete a Group resource.

Every Group resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Group resource. The XDMS shall also ensure that members allocated to a group are also uniquely identified.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Groups".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Groups".

XML Schema

The Group Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in group.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.18.
- Subscriptions document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure the uniqueness of member ids in the group. The XDMS shall also ensure that every identity allocated to a Group resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be three well-known documents. They are as follows:

- The well-known name of the first Group resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents defined under the global tree.

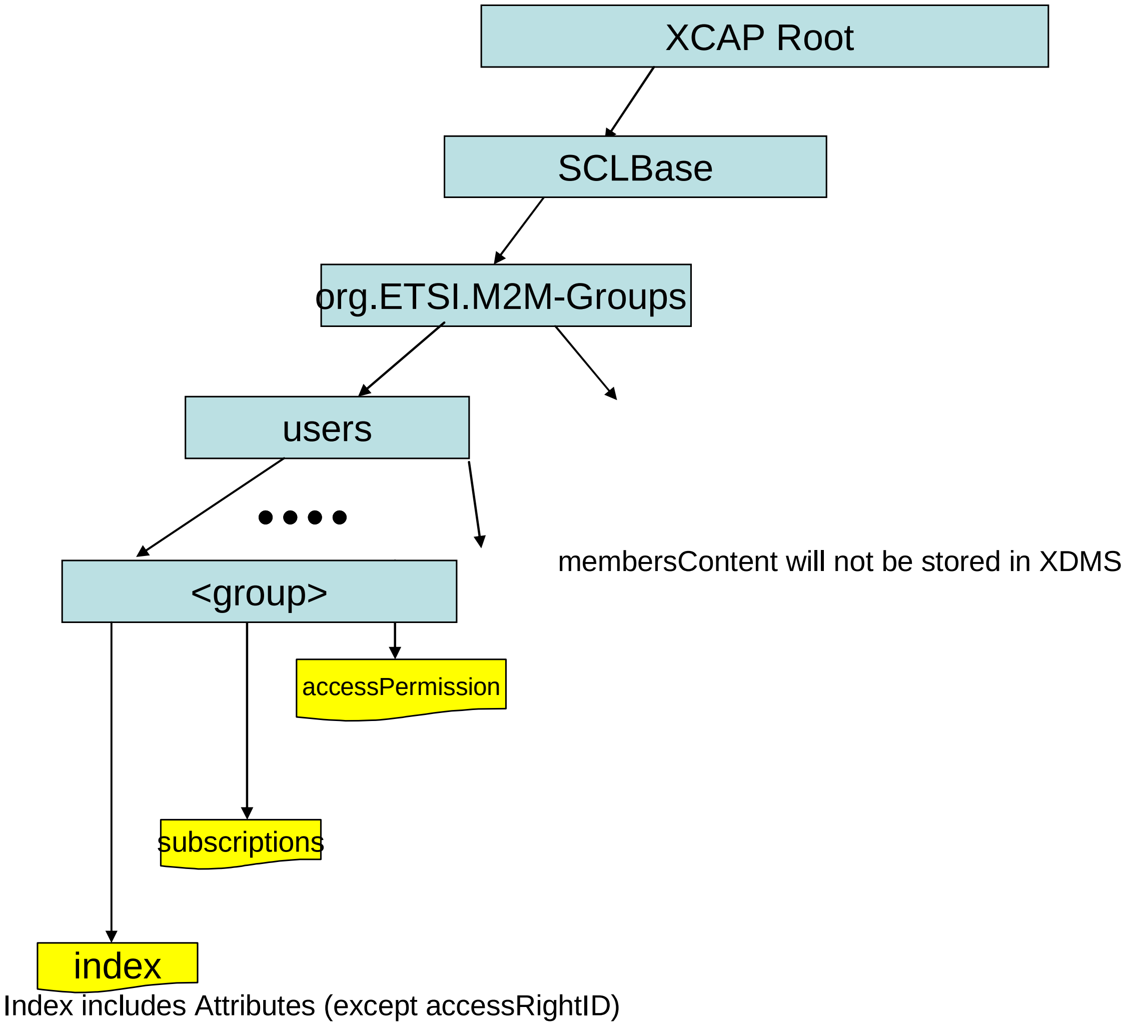

XCAP Root
SCLBase
org.ETSI.M2M-Groups
users
••••
<group>
membersContent will not be stored in XDMS
accessPermission
subscriptions
index
Index includes Attributes (except accessRightID)

Figure F.11: Tree Structure for Group Application Usage

F.1.2.6.2 Mapping between the XDMS XCAP <group> resources and M2M <group> resources

There are multiple locations within the M2M Resource tree structure where <group> resources are defined. As such there are multiple cases for mapping between XDMS XCAP <group> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <group> resources and XDMS XCAP <group> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.14: Group M2M Resource URL Mapping <-> XCAP

M2M <group> Resources	XDMS XCAP <group> Resources
<sclBase>/scls/<scl>/groups/<group>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/groups/<group>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/groups/<group>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 2:

Table F.15: Group M2M Resource URL Mapping <-> XCAP

M2M <group> Resources	XDMS XCAP <group> Resources
<sclBase>/groups/<group>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/groups/<group>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/groups/<group>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 3:

Table F.16: Group M2M Resource URL Mapping <-> XCAP

M2M <group> Resources	XDMS XCAP <group> Resources
<sclBase>/applications/<application>/groups/<group>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>/groups/<group>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/applications/<application>/groups/<group>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 4:

Table F.17: Group M2M Resource URL Mapping <-> XCAP

M2M <group> Resources	XDMS XCAP <group> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups/<group>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups/<group>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/applications/<applicationAnnc>groups/<group>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.6.3 Information Mapping between XDMS XCAP <group> Resources and M2M <group> Resources

Table F.18: Group XDMS Resource <-> M2M Resource

XDMS XCAP <group> Resources	M2M <group> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the group excluding accessRightID attribute and membersContent attribute
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights-/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.7 Management of Containers Resources

F.1.2.7.1 Container Application Usage

The Container Application Usage allows an M2M entity to be able to create/modify/delete a Container resource.

Every Container resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Container resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Containers".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Containers".

XML Schema

The Container Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in container.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.23.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- ContentInstances document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Container resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known documents. They are as follows:

- The well-known name of the first Container resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- Finally, the last well known document shall be "contentInstances". The document selector to access the "contentInstances" XDM document shall be "[auid]/users/[xui]/contentInstances.

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents defined under the global tree.

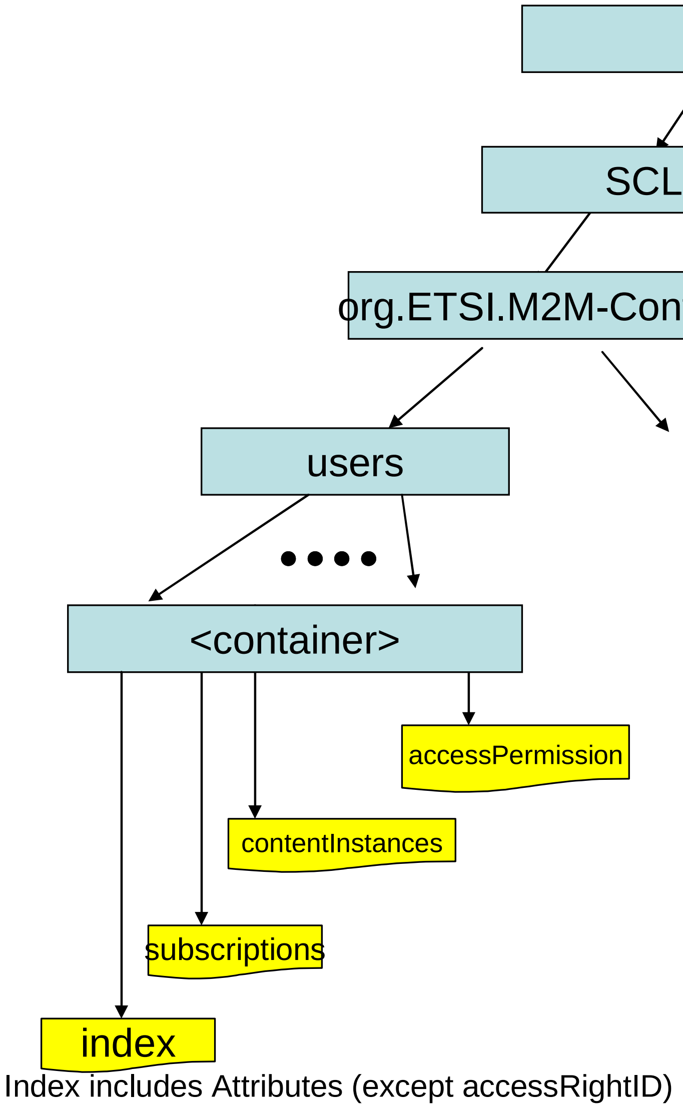


Figure F.12: Tree Structure for Container Application Usage

F.1.2.7.2 Mapping between the XDMS XCAP <container> resources and M2M <container> resources

There are multiple locations within the M2M Resource tree structure where <container> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <container> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <container> resources and XDMS XCAP <container> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.19: Container M2M Resource URL Mapping <-> XCAP

M2M <container> Resources	XDMS XCAP <container> Resources
<sclBase>/scls/<scl>/containers/<container>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/containers/<container>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/scls/<scl>/containers/<container>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/containers/<container>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 2:

Table F.20: Container M2M Resource URL Mapping <-> XCAP

M2M <container> Resources	XDMS XCAP <container> Resources
<sclBase>/containers/<container>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/containers/<container>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/containers/<container>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/containers/<container>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 3:

Table F.21: Container M2M Resource URL Mapping <-> XCAP

M2M <container> Resources	XDMS XCAP <container> Resources
<sclBase>/applications/<application>/containers/<container>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/ applications/<application>/containers/<container>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/ applications/<application>/containers/<container>/subscriptions	XCAPbase//XUI/subscriptions
<sclBase>/applications/<application>/containers/<container>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 4:

Table F.22: Container M2M Resource URL Mapping <-> XCAP

M2M <container> Resources	XDMS XCAP <container> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/<container>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/ applications/<applicationAnnc>/containers/<container>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/scls/<scl>/ applications/<applicationAnnc>/containers/<container>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>applications/<applicationAnnc>/containers/<container>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.7.3 Information Mapping between XDMS XCAP <container> Resources and M2M <container> Resources

Table F.23: Container XDMS Resource <-> M2M Resource

XDMS XCAP <container> Resources	M2M <container> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the container excluding accessRightID attribute.
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/contentInstances	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights-/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.8 Management of Group Collection Resources

F.1.2.8.1 Group Collection Application Usage

The Group Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of Group resources.

Each Group Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Group Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Group-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Group-Collections".

XML Schema

The Group Collections Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in groups.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- Group document, per XUI, under users' tree shall conform to the XML schema as defined in table F.28.
- GroupAnnc, per XUI, document under users' tree shall conform to the XML schema as defined in table F.28.
- AccessPermission per XUI document under users' tree shall conform to an XCAP reference as described in table F.28.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Group Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be five well-known Group Collection XDM documents. They are as follows:

- The well-known name of the first Group Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known document shall be "group". The document selector to access the "group" XDM document shall be "[auid]/users/[xui]/group".
- Finally, the last well-known document shall be "groupAnnc". The document selector to access the "groupAnnc" XDM document shall be "[auid]/users/[xui]/groupAnnc".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

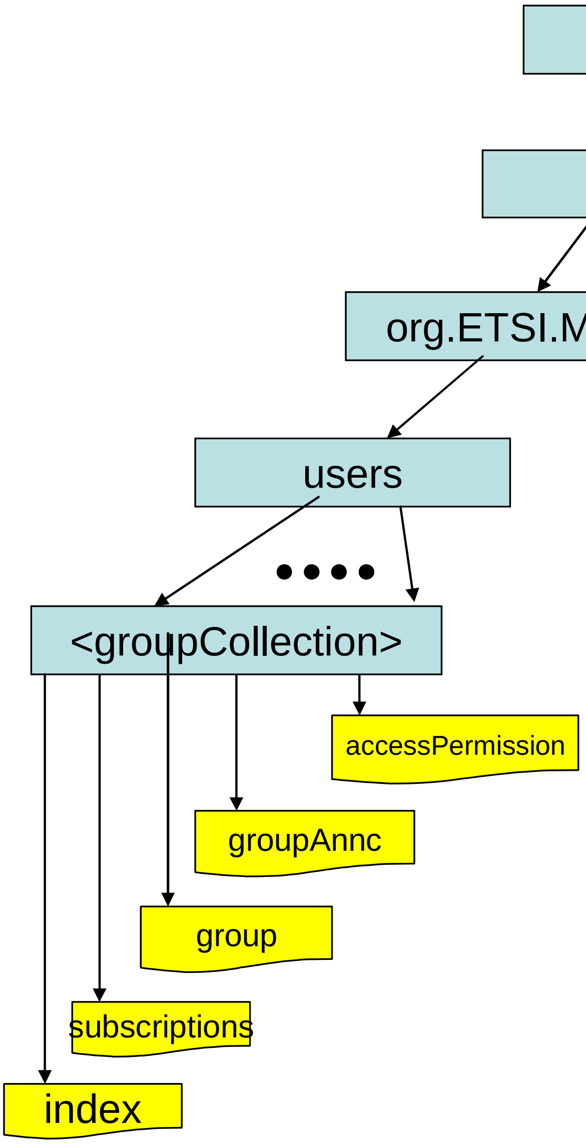


Figure F.13: Tree Structure for Group Collection Application Usage

F.1.2.8.2 Mapping between the XDMS XCAP <groups> resource URL and M2M <groups> resources

There are multiple locations within the M2M Resource tree structure where <groups> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <groups> and M2M resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.24: Group Collection M2M Resource URL Mapping <-> XCAP

M2M <groups> Resources	XDMS XCAP <groups> Resources
<sclBase>/scls/<scl>/ groups/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/ groups/<group>	XCAPbase/XUI/group
<sclBase>/scls/<scl>/groups/<groupAnnc>	XCAPbase/XUI/groupAnnc
<sclBase>/scls/<scl>/groups/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/groups/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 2:

Table F.25: Group Collection M2M Resource URL Mapping <-> XCAP

M2M <groups> Resources	XDMS XCAP <groups> Resources
<sclBase>/groups/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/groups/<group>	XCAPbase/XUI/group
<sclBase>/groups/<groupAnnc>	XCAPbase/XUI/groupAnnc
<sclBase>/groups/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/groups/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 3:

Table F.26: Group Collection M2M Resource URL Mapping <-> XCAP

M2M <groups> Resources	XDMS XCAP <groups> Resources
<sclBase>/applications/<application>/ groups/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>/groups/<group>	XCAPbase/XUI/group
<sclBase>/applications/<application>/groups/<groupAnnc>	XCAPbase/XUI/groupAnnc
<sclBase>/scls/<scl>/applications/application/groups/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/applications/<application>/ groups/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

Case 4:

Table F.27: Group Collection M2M Resource URL Mapping <-> XCAP

M2M <groups> Resources	XDMS XCAP <groups> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/ groups/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups/ <group>	XCAPbase/XUI/group
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups/ <groupAnnc>	XCAPbase/XUI/groupAnnc
<sclBase>/scls/<scl>/applications/<applicationAnnc>/groups/ subscriptions	XCAPbase//XUI/subscriptions
<sclBase>/scls/<scl>/applications/<applicationAnnc>/ groups/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.8.3 Information Mapping Between XDMS XCAP <groups> resource and M2M <groups> resource

Table F.28: Group Collection XDMS Resource <-> M2M Resource

XDMS XCAP <groups> Resources	M2M <groups> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <groups> collection excluding accessRightID attribute
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/group	XDM document includes XCAP references to one or more group document stored under the XUI tree of the Group AUID Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users<group>. The <group> document shall be created and available first before it can be referenced Note that the XDM document may include no XCAP references at all as well, if no groups are defined in this collection
XCAPbase//XUI/groupAnnc	XDM document includes XCAP references to one or more Group Announcement document stored under the XUI tree of the Announced Group AUID Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users<groupAnnc>. The <groupAnnc> document shall be created and available first before it can be referenced Note that the XDM document may include no XCAP references at all as well, if no announced groups are defined in this collection
XCAPbase//XUI/accessPermission	XDM document includes an XCAP reference to an access-right document stored under the XUI tree of the Access Right AUID The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <AccessRight> document shall be created and available first before it can be referenced

F.1.2.9 Management of Container Collection Resources

F.1.2.9.1 Container Collection Application Usage

The Container Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of Container resources.

Each Container Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Container Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Container-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Container-Collections".

XML Schema

The Container Collection resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in containers.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- Container document, per XUI, under users' tree shall conform to table F.33.
- ContainerAnnc document, per XUI, under users' tree shall conform to table F.33.
- LocationContainer document, per XUI, under users' tree shall conform to table F.33.
- LocationContainerAnnc document, per XUI, under users' tree shall conform to table F.33.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described table F.33.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Container Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be seven well-known Container Collection XDM documents. They are as follows:

- The well-known name of the first Container Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known document shall be "locationContainer". The document selector to access the "locationContainer" XDM document shall be "[auid]/users/[xui]/locationContainer".
- The fifth well-known document shall be "locationContainerAnnc". The document selector to access the "locationContainerAnnc" XDM document shall be "[auid]/users/[xui]/locationContainerAnnc".
- The sixth well-known document shall be "container". The document selector to access the "container" XDM document shall be "[auid]/users/[xui]/container".
- Finally, the last well-known document shall be "containerAnnc". The document selector to access the "containerAnnc" XDM document shall be "[auid]/users/[xui]/containerAnnc".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

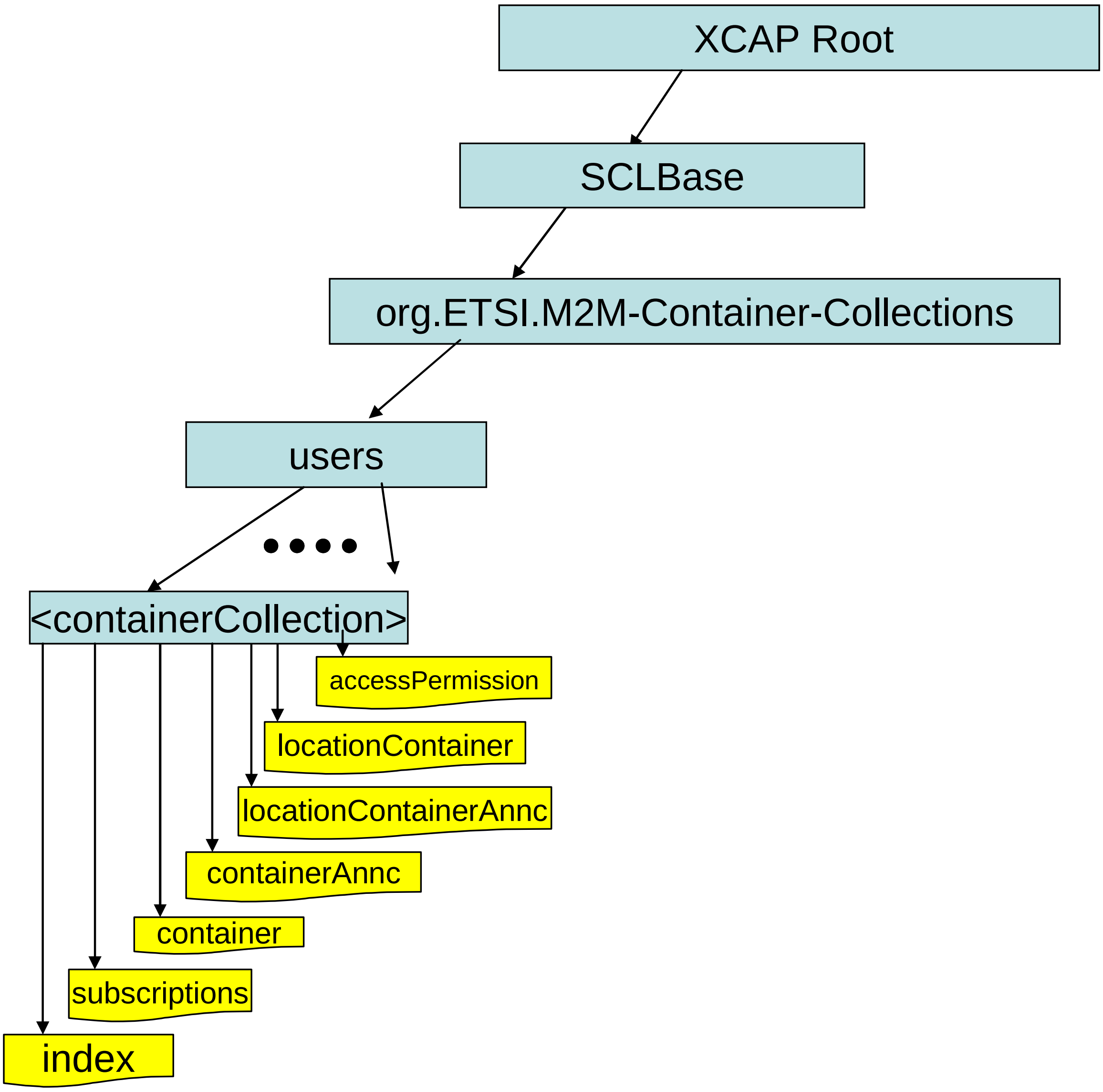


Figure F.14: Tree Structure for Container Collection Application Usage

F.1.2.9.2 Mapping Between XDMS XCAP <containers> resources and M2M <containers> resources

There are multiple locations within the M2M Resource tree structure where <containers> resources are defined. As such there are multiple cases for mapping between XDMS XCAP <containers> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <containers> resources and XDMS XCAP <containers> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.29: Container Collection M2M Resource URL Mapping <-> XCAP

M2M <containers> Resources	XDMS XCAP <containers> Resources
<sclBase>/scls/<scl>/containers/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/containers/<container>	XCAPbase/XUI/container
<sclBase>/scls/<scl>/containers/<containerAnnc>	XCAPbase/XUI/containerAnnc
<sclBase>/scls/<scl>/containers/locationContainer	XCAPbase/XUI/locationContainer
<sclBase>/scls/<scl>/containers/locationContainerAnnc	XCAPbase/XUI/locationContainerAnnc
<sclBase>/scls/<scl>/containers/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/containers/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 2:

Table F.30: Container Collection M2M Resource URL Mapping <-> XCAP

M2M <containers> Resources	XDMS XCAP <containers> Resources
<sclBase>/containers/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/containers/<container>	XCAPbase/XUI/container
<sclBase>/containers/<containerAnnc>	XCAPbase/XUI/containerAnnc
<sclBase>/containers/locationContainer	XCAPbase/XUI/locationContainer
<sclBase>/containers/locationContainerAnnc	XCAPbase/XUI/locationContainerAnnc
<sclBase>/containers/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/containers/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 3:

Table F.31: Container Collection M2M Resource URL Mapping <-> XCAP

M2M <containers> Resources	XDMS XCAP <containers> Resources
<sclBase>/applications/<application>/containers/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>/containers/<container>	XCAPbase/XUI/container
<sclBase>/applications/<application>/containers/<containerAnnc>	XCAPbase/XUI/containerAnnc
<sclBase>/applications/<application>/containers/locationContainer	XCAPbase/XUI/locationContainer
<sclBase>/applications/<appliation>/containers/locationContainerAnnc	XCAPbase/XUI/locationContainerAnnc
<sclBase>/applications/<application>/containers/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/applications/<application>/containers/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 4:

Table F.32: Container Collection M2M Resource URL Mapping <-> XCAP

M2M <containers> Resources	XDMS XCAP <containers> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/<container>	XCAPbase/XUI/container
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/<containerAnnc>	XCAPbase/XUI/containerAnnc
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/locationContainer	XCAPbase/<XUI >/locationContainer
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/locationContainerAnnc	XCAPbase/XUI/locationContainerAnnc
<sclBase>/scls/<scl>/applications/<applicationAnnc >/containers/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers/attribute (accessRightID)	XCAPbase/XUI/accessPermission

F.1.2.9.3 Information Mapping Between XDMS XCAP <containers> resource and M2M <containers> resource

Table F.33: Container Collection XDMS Resource <-> M2M Resource

XDMS XCAP< containers> Resources	M2M <containers> Resources
XCAPbase/XUI/Index	XDM document conforms to the XML schema for all attributes defined for the <containers> collection excluding accessRightID attribute
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/container	XDM document includes XCAP references to one or more container document stored under the XUI tree of the Container AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users<container>. The <container> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no containers are defined in this collection
XCAPbase/XUI/containerAnnc	XDM document includes XCAP references to one or more Container Announcement document stored under the XUI tree of the Announced Container AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Containers/users<containerAnnc>. The <containerAnnc> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no announced containers are defined in this collection
XCAPbase/XUI/locationContainer	XDM document includes XCAP references to one or more Location Container document stored under the XUI tree of the Location Container AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Location-Containers/users<locationContainer>. The <locationContainer> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no location containers are defined in this collection

XDMS XCAP< containers> Resources	M2M <containers> Resources
XCAPbase/XUI/locationContainerAnnc	XDM document includes XCAP references to one or more Location Container Announcement document stored under the XUI tree of the Announced Location Container AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-LocationContainers/users<locationContainerAnnc>. The <locationContainerAnnc> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no announced location containers are defined in this collection
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an access-right document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <AccessRight> document shall be created and available first before it can be referenced

F.1.2.10 Management of AccessRight Collection Resources

F.1.2.10.1 AccessRight Collection Application Usage

The AccessRight Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of AccessRight resources.

Each AccessRight collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Access-Right Collection resource.

The AUID SHALL be "org.ETSI.M2M-AccessRight-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M- AccessRight-Collection".

XML Schema

The Access-Right Collection resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in accessRights.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- AccessRight document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.38.
- AccessRightAnnc per XUI document under users' tree shall conform to an XCAP reference as described in table F.38.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.38.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an AccessRight Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be five well-known AccessRight Collection XDM documents. They are as follows:

- The well-known name of the first AccessRight Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known document shall be "accessRight". The document selector to access the "accessRight" XDM document shall be "[auid]/users/[xui]/accessRight".
- Finally, the last well-known document shall be "accessRightAnnc". The document selector to access the "accessRightAnnc" XDM document shall be "[auid]/users/[xui]/accessRightAnnc".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermisssion document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

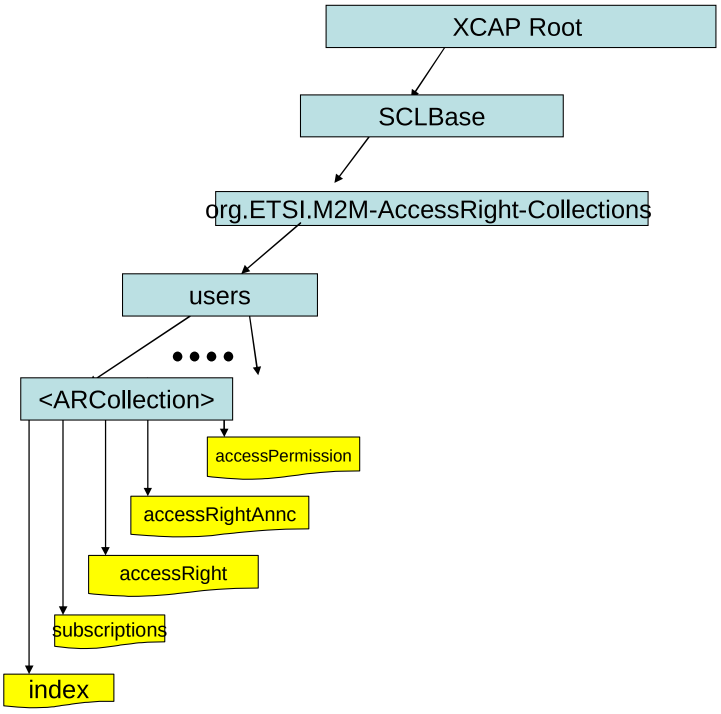


Figure F.15: Tree Structure for accessRight Collection Application Usage

F.1.2.10.2 Mapping between the XDMS XCAP <accessRights> resources and M2M <accessRights> resources

There are multiple locations within the M2M Resource tree structure where <accessRights> resources are defined. As such there are multiple cases for mapping between XDMS XCAP <accessRights> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <accessRights> resources and XDMS XCAP <accessRights> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.34: AccessRight Collection M2M Resource URL Mapping <-> XCAP

M2M <accessRights> Resources	XDMS XCAP <accessRights> Resources
<sclBase>/scls/<scl>/accessRights/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/accessRights/<accessRight>	XCAPbase/XUI/accessRight
<sclBase>/scls/<scl>/accessRights/<accessRightAnnc>	XCAPbase/XUI/accessRightAnnc
<sclBase>/scls/<scl>/accessRights/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/accessRights/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 2:

Table F.35: AccessRight Collection M2M Resource URL Mapping <-> XCAP

M2M <accessRights> Resources	XDMS XCAP <accessRights> Resources
<sclBase>/accessRights/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/accessRights/<accessRight>	XCAPbase/XUI/accessRight
<sclBase>/accessRights/<accessRightAnnc>	XCAPbase/XUI/accessRightAnnc
<sclBase>/accessRights/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/accessRights/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 3:

Table F.36: AccessRight Collection M2M Resource URL Mapping <-> XCAP

M2M <accessRights> Resources	XDMS XCAP <accessRights> Resources
<sclBase>/applications/<application>/accessRights/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>/accessRights/<access Right>	XCAPbase/XUI/accessRight
<sclBase>/applications/<application>/accessRights/<access RightAnnc>	XCAPbase/XUI/accessRightAnnc
<sclBase>/applications/<application>/ accessRights/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/applications/<application>/accessRights/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 4:

Table F.37: AccessRight Collection M2M Resource URL Mapping <-> XCAP

M2M <accessRights> Resources	XDMS XCAP <accessRights> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/<accessRight>	XCAPbase/XUI/accessRight
<sclBase>/scls/<scl>/applications/<applicationAnnc/>accessRights/<accessRightAnnc>	XCAPbase/XUI/accessRightAnnc
<sclBase>/scls/<scl>/applications/<applicationAnnc>/ accessRights/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights/attribute (accessRightID)	XCAPbase/XUI/accessPermission

F.1.2.10.3 Information Mapping between XDMS XCAP <accessRights> resources and M2M <accessRights> resources

Table F.38: AccessRight Collection XDMS Resource <-> M2M Resource

XDMS XCAP <accessRights> Resource	M2M <accessRights> resource
XCAPbase/XUI/Index/attributes	The XDM document conforms to the XML schema for all attributes defined for the <accessRights> collection excluding accessRightID attribute
XCAPbase/<ARCollection>/subscriptions	Empty Document
XCAPbase/<ARCollection>/accessPermission	XDM document includes an XCAP reference to an access-right document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be created and available first before it can be referenced
XCAPbase/<ARCollection>/accessRight	XDM document includes XCAP references to one or more Access Right document stored under the XUI tree of the Access Right AUID. Each XCAP reference will have the following URL: XCAPbase<accessRight>. The <accessRight> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no access Rights are defined in this collection
XCAPbase/<ARCollection>/accessRightAnnc	XDM document includes XCAP references to one or more AccessRight Announcement document stored under the XUI tree of the Announced AccessRights AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-AccessRights/users<accessRightAnnc>. The <accessRightAnnc> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no announced groups are defined in this collection

F.1.2.11 Management of Application Collection Resources

F.1.2.11.1 Application Collection Application Usage

The Application Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of locally registered applications as well as announced-Application resources.

Each Applications Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Applications Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Application-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Applications-Collections".

XML Schema

The Applications Collections resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in applications.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Application document, per XUI, under users' tree shall conform to table F.41.
- ApplicattionAnnc document, per XUI, under users' tree shall conform to table F.41.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference URI as described in table F.41.
- Subscriptions document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an Application Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be five well-known documents. They are as follows:

- The well-known name of the main Application Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known document shall be "application". The document selector to access the "application" XDM document shall be "[auid]/users/[xui]/application".

- Finally, the last well-known document shall be "applicationAnnc". The document selector to access the "applicationAnnc" XDM document shall be "[auid]/users/[xui]/applicationAnnc".

There is a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

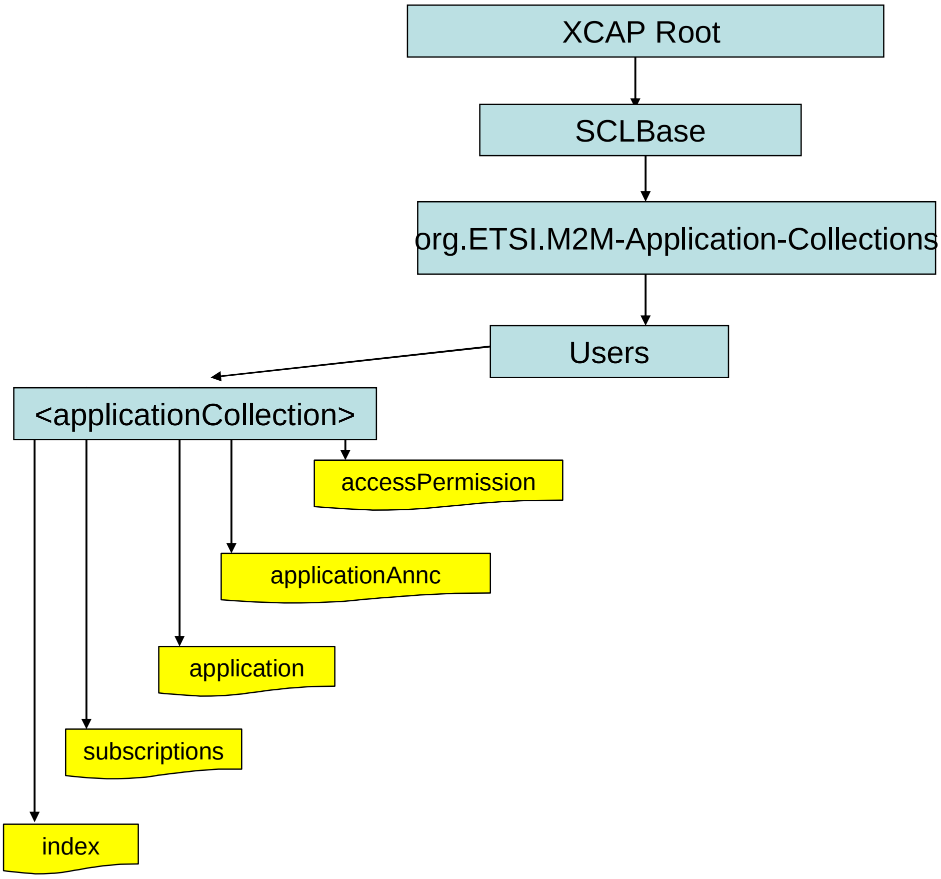


Figure F.16: Tree Structure for Application Collection Application Usage

F.1.2.11.2 Mapping between the XDMS XCAP <applications> resources and M2M <applications> resources

There are two locations within the M2M Resource tree structure where <applications> resources are defined. As such, there are two potential cases for mapping between XDMS XCAP <applications> and M2M resources.

Furthermore it is to be noted that a1:1 mapping exists between M2M <applications> resources and XDMS XCAP <applications> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Applications-Collections/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.39: Application Collection M2M Resource URL Mapping <-> XCAP

M2M <applications> Resource	XDMS XCAP <applications> Resource
<sclBase>/applications/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/applications/<application>	XCAPbase/XUI/application
<sclBase>/applications/<applicationAnnc>	XCAPbase/XUI/applicationAnnc
<sclBase>/applications/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/applications/attribute (accessRightID)	XCAPbase/XUI/ accessPermission

Case 2:

Table F.40: Application Collection M2M Resource URL Mapping <-> XCAP

M2M <applications> Resources	XDMS XCAP <applications> Resources
<sclBase>/scls/<scl>/applications/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/applications/<application>	XCAPbase/XUI/application
<sclBase>/scls/<scl>/applications><applicationAnnc>	XCAPbase/XUI/applicationAnnc
<sclBase>/scls/<scl>/applications/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/applications/attribute (accessRightID)	XCAPbase/XUI/ accessPermission

F.1.2.11.3 Information Mapping between XDMS XCAP <applications> resource and M2M <applications> resource

Table F.41: Application Collection XDMS Resource <-> M2M Resource

XDMS XCAP <applications> Collection Resource	M2M <applications>Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the application collection excluding accessRightID attribute
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/application	XDM document includes XCAP references to one or more application document stored under the XUI tree of the Local Application AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users<application>. The <application> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no applications are defined in this collection
XCAPbase/XUI/applicationAnnc	XDM document includes XCAP references to one or more Application Announcement document stored under the XUI tree of the Announced Application AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users<applicationAnnc>. The <applicationAnnc> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no applicationAnnc are defined in this collection
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an access-right document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Right/users<accessRight>. The <accessRight> document shall be created and available first before it can be referenced

F.1.2.12 Management of SCL Collection Resource

F.1.2.12.1 SCL Collection Application Usage

The SCL Collection Application Usage allows an M2M entity to be able to create/modify/delete an SCL collection.

There shall be a single SCL collection created and it shall be allocated a unique identity and shall be located below the users' tree located beneath the AUID tree allocated to SCL collection resources.

Application Unique ID

The AUID SHALL be "org..ETSI.M2M-SCL-Collection".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-SCL-Collection".

XML Schema

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in scls.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP URI as defined in table F.43.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an SCL Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be three well-known documents. They are as follows:

- The well-known name of the main SCL collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- Finally, the last well-known document shall be shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

Given that there is a single SCL collection and all registered scls are implicitly members in this collection, there will not be a specific XDM document that holds members in the SCL collection. Rather all registered scls in the Registered SCL Application Usage are members in the SCL collection.

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents for this application usage.

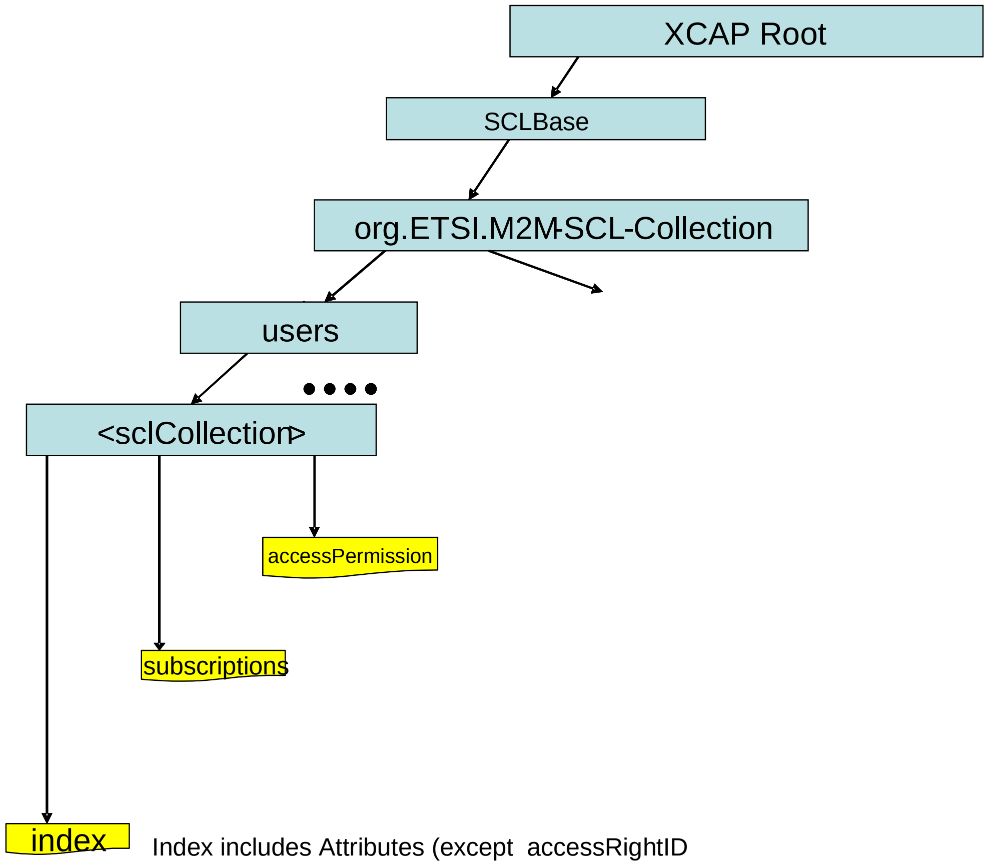

XCAP Root
SCLBase
org.ETSI.M2M-SCL-Collection
users
●●●●
<sclCollection>
accessPermission
subscriptions
index
Index includes Attributes (except accessRightID

Figure F.17: Tree Structure for SCL Collection Application Usage

F.1.2.12.2 Mapping between XDMS XCAP <scls> resources and M2M <scls> resources

It is to be noted that a 1:1 mapping exists between M2M <scls> resources and XDMS XCAP <scls> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/.

Table F.42: SCL Collection M2M Resource URL Mapping <-> XCAP

M2M <scls> Resource URL	XDMS XCAP <scls> Resource URL
<sclBase>/scls/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/attribute
<sclBase>/scls/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/attribute (AccessRightID)	XCAPbase/XUI/accessPermission

F.1.2.12.3 Information Mapping between the XDMS XCAP <scls> resources and M2M <scls> resources

Table F.43: SCL Collection XDMS Resource <-> M2M Resource

XDMS XCAP <scls> Resource	M2M <scls> Resource
XCAPbase/XUI/attributes	XDM document conforms to the XML schema for all attributes defined for the SCL collection excluding accessRightID attribute
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot//<sclBase>/org.ETSI.M2M-Acess-Right/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.13 Management of Location Containers Resources

F.1.2.13.1 Location Container Application Usage

The Location Container Application Usage allows an M2M entity to be able to create/modify/delete a Location Container resource.

Every Location Container resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Location Container resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Location-Containers".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Location-Containers".

XML Schema

The Location Container Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in locationContainer.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as depicted in table F.48.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- ContentInstances document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Location Container resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known documents. They are as follows:

- The well-known name of the first Location Container resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well known document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The third well known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- Finally, the last well known document shall be "contentInstances". The document selector to access the "contentInstances" XDM document shall be "[auid]/users/[xui]/contentInstances.

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents defined under the global tree.

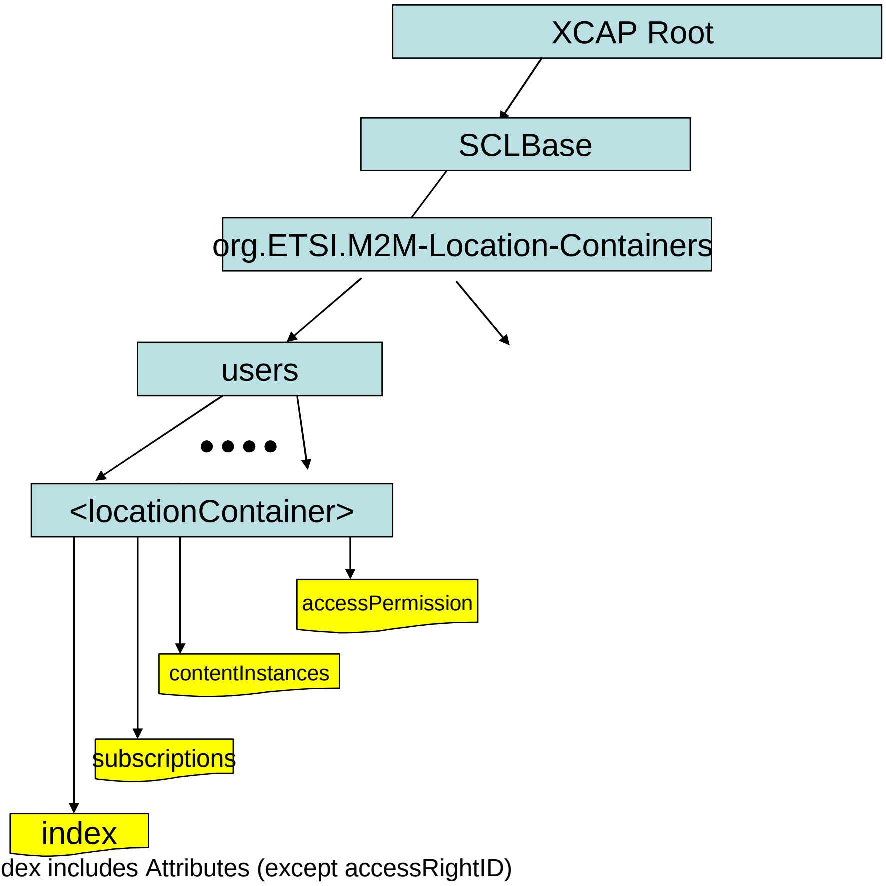


Figure F.18: Tree Structure for Location Container Application Usage

F.1.2.13.2 Mapping between the XDMS XCAP <locationContainer> resources and M2M <locationContainer> resources

There are multiple locations within the M2M Resource tree structure where <locationContainer> resources are defined. As such ,there are multiple cases for mapping between XDMS XCAP <locationContainer> and M2M resources.

It is to be noted that a 1:1 mapping exists between M2M <locationContainer> resources and XDMS XCAP <locationContainer> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Location-Containers/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Case 1:

Table F.44: Location Container M2M Resource URL Mapping <-> XCAP

M2M <locationContainer> Resources	XDMS XCAP <locationContainer> Resources
<sclBase>/scls/<scl>/containers/<locationContainer>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/containers/<locationContainer>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/scls/<scl>/containers/<locationContainer>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/containers/<locationContainer>/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 2:

Table F.45: Location Container M2M Resource URL Mapping <-> XCAP

M2M <locationContainer> Resources	XDMS XCAP <locationContainer> Resources
<sclBase>/containers/<locationContainer>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/containers/<locationContainer>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/containers/<locationContainer>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/containers/<locationContainer>/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 3:

Table F.46: Location Container M2M Resource URL Mapping <-> XCAP

M2M <locationContainer> Resources	XDMS XCAP <locationContainer> Resources
<sclBase>/applications/<application>/containers/<locationContainer>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/ applications/<application>/containers/<locationContainer>/contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/ applications/<application>/containers/<locationContainer>/subscriptions	XCAPbase//XUI/subscriptions
<sclBase>/applications/<application>/containers/<locationContainer>/attribute (accessRightID)	XCAPbase/XUI/accessPermission

Case 4:

Table F.47: Location Container M2M Resource URL Mapping <-> XCAP

M2M <locationContainer> Resources	XDMS XCAP <locationContainer> Resources
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers /<locationContainer>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/ applications/<applicationAnnc>/containers/<locationContainer>/ contentInstances	XCAPbase/XUI/contentInstances
<sclBase>/scls/<scl>/ applications/<applicationAnnc>/containers/<locationContainer>/ subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers /<locationContainer>/attribute (accessRightID)	XCAPbase/XUI/accessPermission

F.1.2.13.3 Information Mapping between XDMS XCAP <locationContainer> Resources and M2M <locationContainer> Resources

Table F.48: Location Container XDMS Resource <-> M2M Resource

XDMS XCAP <locationContainer> Resources	M2M <locationContainer> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the Location Container excluding accessRightID attribute in additions to membersResource
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/contentInstances	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights-/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.14 Management of Announced Group resources

F.1.2.14.1 M2M Announced Groups Application Usage

The M2M Announced Groups Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an M2M Announced Groups resources.

Every M2M Announced Group resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M Announced Groups resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Announced-Groups".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Announced-Groups".

XML Schema

The M2M Announced Groups XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in groupAnnc.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference described as per table F.50.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M Announced Groups resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents. They are as follows:

- The well-known name of the main M2M Announced Groups resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

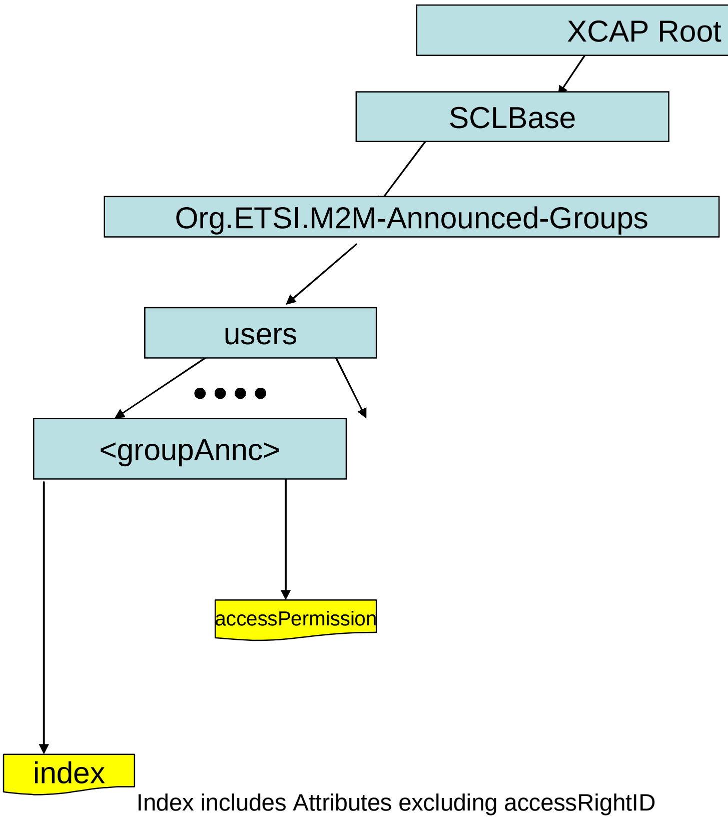

XCAP Root
SCLBase
Org.ETSI.M2M-Announced-Groups
users
••••
<groupAnnc>
accessPermission
index
Index includes Attributes excluding accessRightID

Figure F.19: Tree Structure for Group Announcement Application Usage

F.1.2.14.2 Mapping between the XDMS XCAP <groupAnnc> resources and M2M <groupAnnc> resources

There are multiple locations within the M2M Resource tree structure where <groupAnnc> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <groupAnnc> and M2M resources. However only one case shall be shown here.

It is to be noted that a 1:1 mapping exists between M2M <groupAnnc> resources and XDMS XCAP <groupAnnc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.49: Announced Group M2M Resource URL Mapping <-> XCAP URL

M2M <groupAnnc> Resources	XDMS XCAP <groupAnnc> Resources
<sclBase>/scls/<scl>/groups/<groupAnnc>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls<scl>/groups/<groupAnnc>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.14.3 Information Mapping Between XDMS XCAP <groupAnnc> resources and M2M <groupAnnc> resources

Table F.50: Announced Groups XDMS Resource <-> M2M Resource

XDMS XCAP <groupAnnc> Resource	M2M <groupAnnc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <groupAnnc> resource except accessRightID
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.15 Management of Announced Containers resources

F.1.2.15.1 M2M Announced Containers Application Usage

The M2M Announced Containers Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an M2M Announced Containers resources.

Every M2M Announced Container resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M Announced Containers resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Announced-Containers".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Announced-Containers".

XML Schema

The M2M Announced Containers XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in containerAnnc.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.

- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.52.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M Announced Containers resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents. They are as follows:

- The well-known name of the main M2M Announced Containers resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

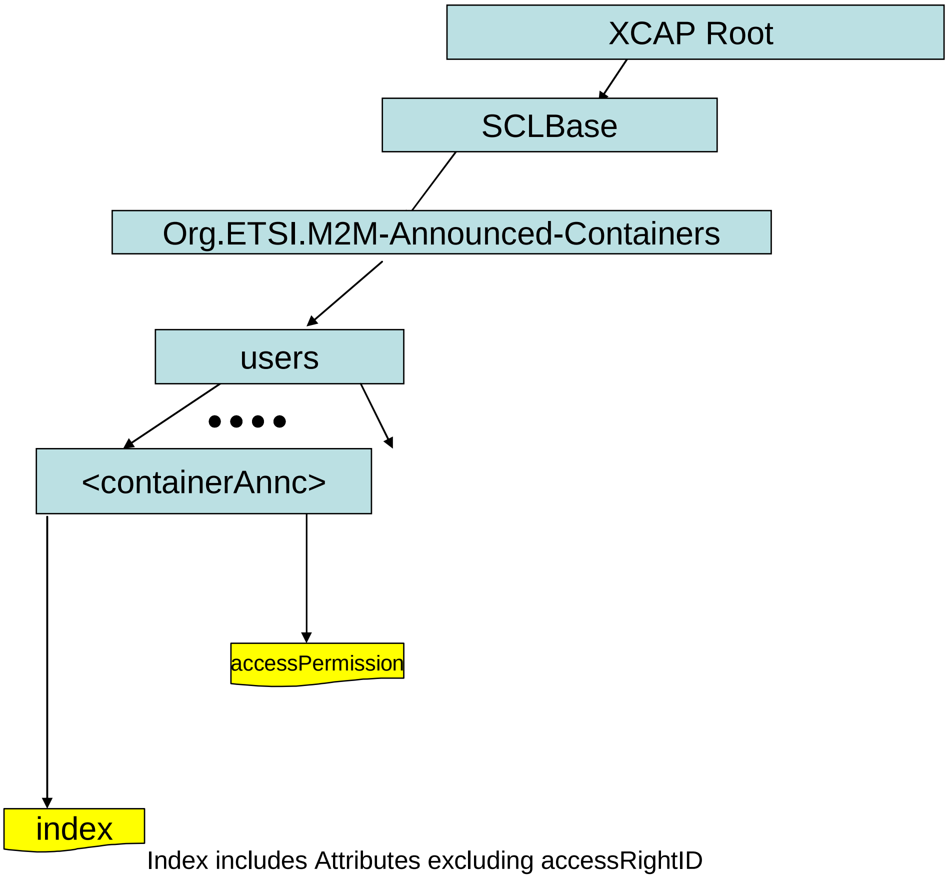
There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.



Index includes Attributes excluding accessRightID

Figure F.20: Tree Structure for ContainerAnnouncement Application Usage

F.1.2.15.2 Mapping between the XDMS XCAP <containerAnnc> resources and M2M <containerAnnc> resources

There are multiple locations within the M2M Resource tree structure where <containerAnnc> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <containerAnnc> and M2M <containerAnnc> resources. However only one case will be described here.

It is to be noted that a 1:1 mapping exists between M2M <containerAnnc> resources and XDMS XCAP <containerAnnc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Containers/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.51: Announced Containers M2M Resource URL Mapping <-> XCAP URL

M2M <containerAnnc> Resources	XDMS XCAP <containerAnnc> Resources
<sclBase>/scls/<scl>/containers/<containerAnnc>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls<scl>/containers/<containerAnnc>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.15.3 Information Mapping Between XDMS XCAP <containerAnnc> resources and M2M <containerAnnc> resources

Table F.52: Announced Containers XDMS Resource <-> M2M Resource

XDMS XCAP <containerAnnc> Resource	M2M <containerAnnc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <containerAnnc> resource except accessRightID
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.16 Management of Announced Access Rights resources

F.1.2.16.1 M2M Announced Access Right Application Usage

The M2M Announced Access Rights Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an M2M Announced accessRights resources.

Every M2M Announced accessRight resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M Announced accessRight resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Announced- AccessRights".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Announce-AccessRights".

XML Schema

The M2M Announced Access Rights XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in accessRightAnnc.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described below in table F.54.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M Announced accessRight resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents. They are as follows:

- The well-known name of the main M2M Announced accessRight resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".

- The second well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

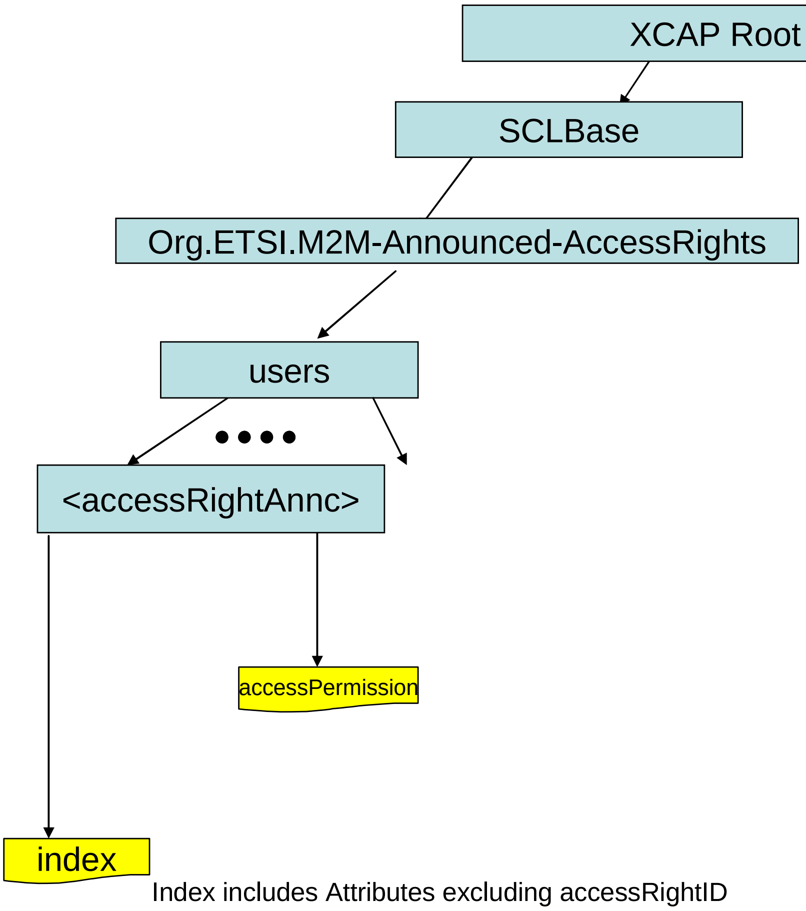
There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.



Index includes Attributes excluding accessRightID

Figure F.21: Tree Structure for AccessRight Announcement Application Usage

F.1.2.16.2 Mapping between the XDMS XCAP <accessRightAnnc> resources and M2M <accessRightAnnc> resources

There are multiple locations within the M2M Resource tree structure where <accessRightAnnc> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <accessRightAnnc> and M2M <accessRightAnnc> resources. However, only one case will be shown here.

It is to be noted that a 1:1 mapping exists between M2M <accessRightAnnc> resources and XDMS XCAP <accessRightAnnc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-AccessRights/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.53: Announced accessRight M2M Resource URL Mapping <-> XCAP URL

M2M <accessRightAnnc> Resources	XDMS XCAP <accessRightAnnc> Resources
<sclBase>/scls/<scl>/containers/<accessRightAnnc>/attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls<scl>/containers/<accessRightAnnc>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.16.3 Information Mapping Between XDMS XCAP <accessRightAnnc> resources and M2M <accessRightAnnc> resources

Table F.54: Announced Containers XDMS Resource <-> M2M Resource

XDMS XCAP <accessRightAnnc> Resource	M2M <accessRightAnnc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <accessRightAnnc> resource except accessRightID
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.17 Management of Announced Location Containers resources

F.1.2.17.1 M2M Announced Location Containers Application Usage

The M2M Announced Location Container Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an M2M Announced Location Containers resources.

Every M2M Announced Location Containers resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M Announced Location Containers resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Announced-LocationContainers".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Announced-LocationContainers".

XML Schema

The M2M Announced locationContainers XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in locationContainerAnnc.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.56.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M Announced locationContainers resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents. They are as follows:

- The well-known name of the main M2M Announced locationContainer resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

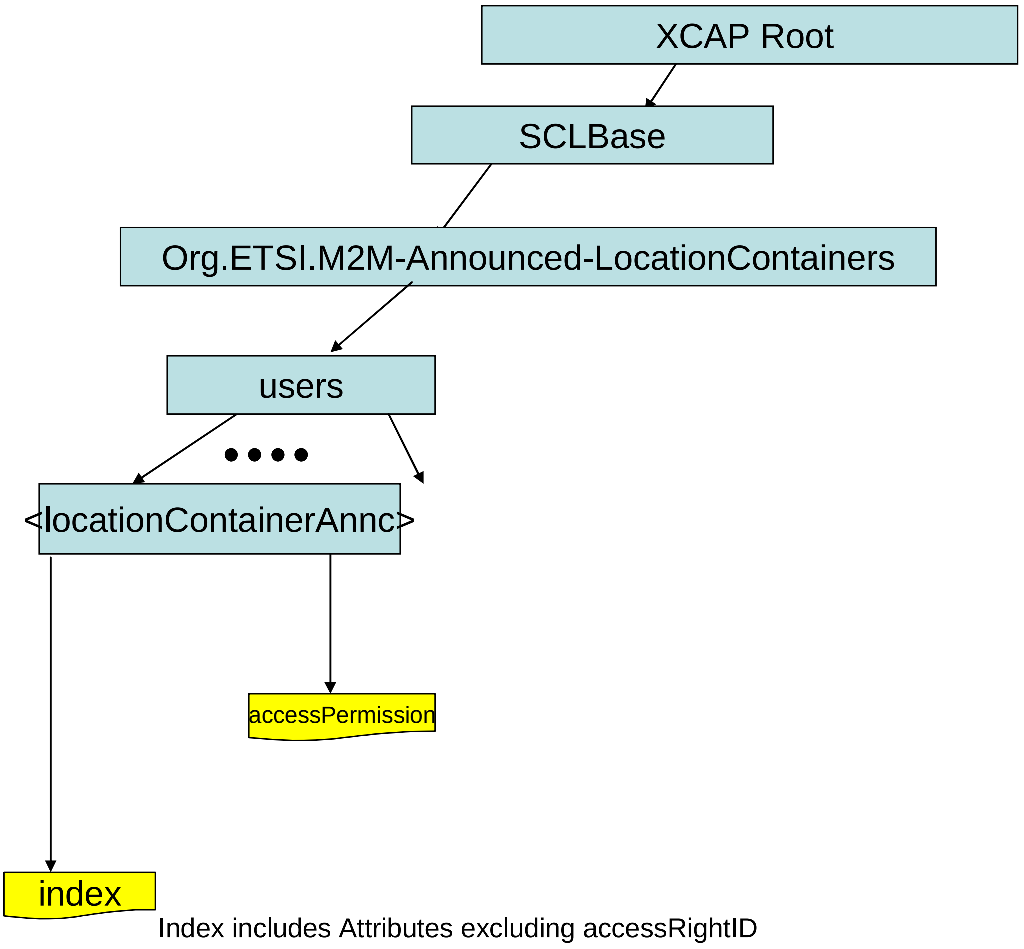

XCAP Root
SCLBase
Org.ETSI.M2M-Announced-LocationContainers
users
<locationContainerAnnc>
accessPermission
index
Index includes Attributes excluding accessRightID

Figure F.22: Tree Structure for Location ContainerAnnouncement Application Usage

F.1.2.17.2 Mapping between the XDMS XCAP <locationContainerAnnc> resources and M2M <locationContainerAnnc> resources

There are multiple locations within the M2M Resource tree structure where <locationContainerAnnc> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <location ContainerAnnc> and M2M <locationContainerAnnc> resources. However, only one case will be shown here.

It is to be noted that a 1:1 mapping exists between M2M <locationContainerAnnc> resources and XDMS XCAP <locationContainerAnnc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-LocationContainers/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.55: Announced Location Containers M2M Resource URL Mapping <-> XCAP URL

M2M <locationContainerAnnc> Resources	XDMS XCAP <locationContainerAnnc> Resources
<sclBase>/scls/<scl>/containers/<locationContainerAnnc>/ attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls<scl>/containers/<locationContainerAnnc>/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.1.2.17.3 Information Mapping Between XDMS XCAP <locationContainerAnnc> resources and M2M <locationContainerAnnc> resources

Table F.56: Announced locationContainers XDMS Resource <-> M2M Resource

XDMS XCAP <locationContainerAnnc> Resource	M2M <locationContainerAnnc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <locationContainerAnnc> resource except accessRightID
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Acess-Rights/users<accessRight>. The <accessRight> shall be first created or available so its reference can be stored here

F.1.2.18 Management of ContentInstance Collection resources

F.1.2.18.1 ContentInstances Collection Application Usage

The contentInstance Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of contentInstance resources.

Each contentInstance Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the contentInstance Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-ContentInstance-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-ContentInstance-Collections".

XML Schema

The contentInstance Collections Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in contentInstances.xsd, with the root element the sclBase, with the exception of child references.
- ContentInstance document, per XUI, under users' tree shall conform to table F.58.

- Subscriptions document under users' tree, per XUI, shall be an empty document.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.58.
- Latest document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.58.
- Oldest document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.58.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a contentInstance Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be six well-known contentInstance Collection XDM documents. They are as follows:

- The main well-known name of the contentInstance Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/ index".
- The second well-known name of the contentInstance Collection resource XDM document shall be "contentInstance". The document selector to access the "contentInstance" XDM document shall be "[auid]/users/[xui]/contentInstance".
- The third well-known name of the contentInstance Collection resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known name of the contentInstance Collection resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/ subscriptions".
- The fifth well-known name of the contentInstance Collection resource XDM document shall be "latest". The document selector to access the "latest" XDM document shall be "[auid]/users/[xui]/ latest".
- The last well-known name of the contentInstance Collection resource XDM document shall be "oldest". The document selector to access the "oldest" XDM document shall be "[auid]/users/[xui]/ oldest".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

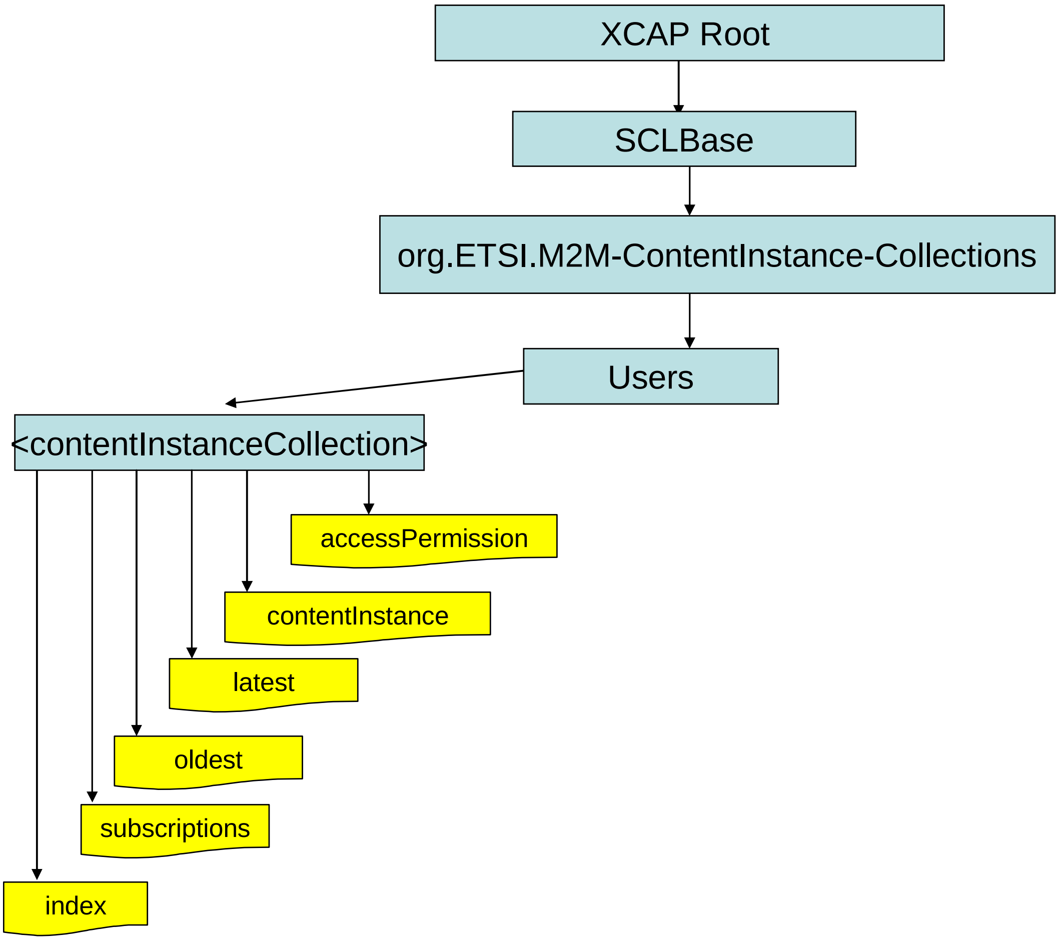


Figure F.23: Tree Structure for contentInstance Collection Application Usage

F.1.2.18.2 Mapping between the XDMS XCAP <contentInstances> resource URL and M2M < contentInstances> resources

There are multiple locations within the M2M Resource tree structure where <contentInstances> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <contentInstances> and M2M<contentInstances> resources. However, only one case shall be described here.

The XCAP base for this application usage is XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.57: ContentInstance Collection M2M Resource URL Mapping <-> XCAP

M2M <contentInstances> Resources	XDMS XCAP <contentInstances> Resources
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ attribute	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ <contentInstance>	XCAPbase/XUI/contentInstance
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ subscriptions	XCAPbase/XUI/subscriptions
NSCL sets the permission to the parent value by reading first the value allocated to the parent and then setting it here.	XCAPbase/XUI/accessPermission
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ latest	XCAPbase/XUI/latest
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ oldest	XCAPbase/XUI/oldest

F.1.2.18.3 Information Mapping Between XDMS XCAP <contentInstances> resource and M2M <contentInstances> resource

Table F.58: ContentInstances Collection XDMS Resource <-> M2M Resource

XDMS XCAP <contentInstances> Resources	M2M <contentInstances> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <contentInstances> collection
XCAPbase/XUI/contentInstance	XDM document includes XCAP references to one or more contentInstance document stored under the XUI tree of the contentInstance AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users<contentInstance>. The <contentInstance> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no contentInstance is defined in this collection
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot//<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here
XCAPbase/XUI/latest	XDM document includes a reference to the actual M2M URI for the actual contentInstance holding information. In this case, the M2M NSCL has to read the present document to fetch the reference to the requested document, and then it has to execute a second read to read the actual data itself from that reference
XCAPbase/XUI/oldest	XDM document includes a reference to the actual M2M URI for the actual contentInstance holding information. In this case, the M2M NSCL has to read the present document to fetch the reference to the requested document, and then it has to execute a second read to read the actual data itself from that reference

F.1.2.19 Management of ContentInstance resources

F.1.2.19.1 Content Instance Application Usage

The M2M contentInstance Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read M2M contentInstance resources.

Every M2M contentInstances resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M contentInstance resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Content-Instances".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Content-Instances".

XML Schema

The M2M contentInstance XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in contentInstance.xsd, with the root element the sclBase, with the exception of child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.60.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M contentInstance resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents. They are as follows:

- The well-known name of the main M2M contentInstance resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the M2M contentInstance resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

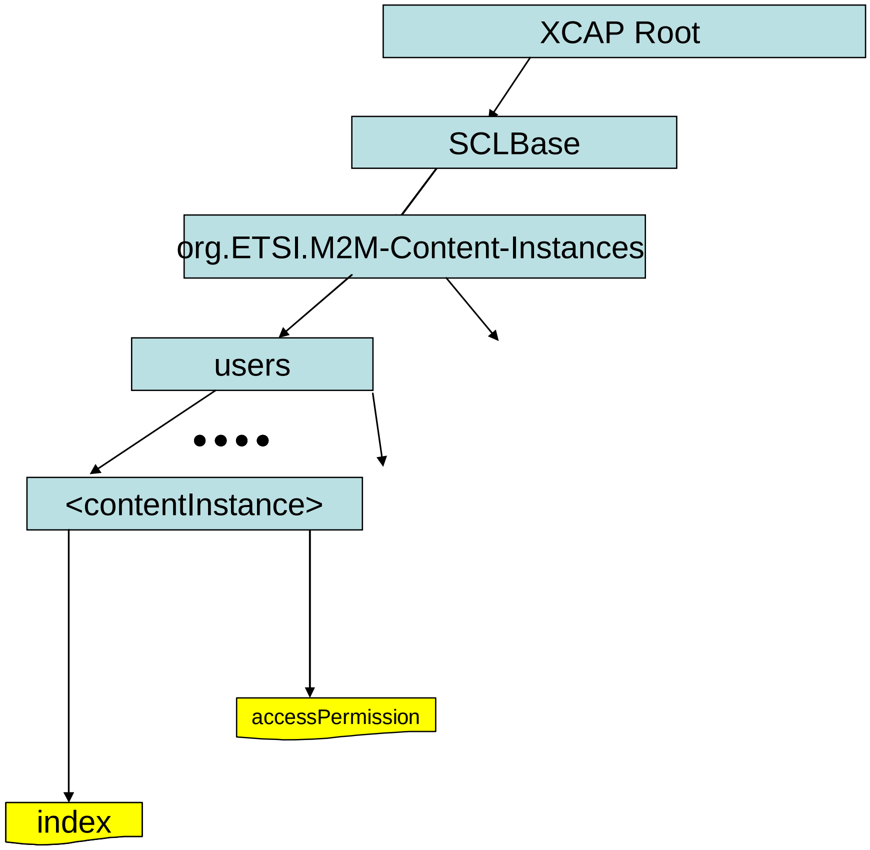


Figure F.24: Tree Structure for contentInstance Application Usage

F.1.2.19.2 Mapping between the XDMS XCAP <contentInstance> resources and M2M <contentInstance> resources

There are multiple locations within the M2M Resource tree structure where <contentInstance> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <contentInstance> and M2M <contentInstance> resources. However, only one case shall be described here.

It is to be noted that a 1:1 mapping exists between M2M <contentInstance> resources and XDMS XCAP <contentInstance> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-m2m2pocs/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.59: ContentInstance M2M Resource URL Mapping <-> XCAP URL

M2M <contentInstance> Resources	XDMS XCAP <contentInstance> Resources
<sclBase>/scls/<scl>/ containers/<container>/contentInstances/ <contentInstance>/attribute	XCAPbase/XUI/index
NSCL shall set the permission to the proper values	XCAPbase/XUI/accessPermission

F.1.2.19.3 Information Mapping Between XDMS XCAP <contentInstance> resources and M2M <contentInstance> resources

Table F.60: ContentInstance XDMS Resource <-> M2M Resource

XDMS XCAP <contentInstance> Resource	M2M <contentInstance> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <contentInstance> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.2.20 Management of Subscriptions Collection resources

F.1.2.20.1 Subscription Collection Application Usage

The Subscription Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of Subscription resources.

Each Subscription Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the Subscription Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Subscription-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Subscription-Collections".

XML Schema

The Subscription Collections Resource XDM documents shall conform to the following XML schemas:

- Subscription document, per XUI , under users' tree shall conform to XML schema as defined in table F.62.
- AccessPermission per XUI document under users' tree shall conform to an XCAP reference as defined in table F.62.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a Subscription Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known Subscription Collection XDM documents. They are as follows:

- The first well-known name of the Subscription Collection resource XDM document shall be "subscription". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/ subscription".

- The last well-known name of the Subscription Collection resource XDM document shall be "accessPermission". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

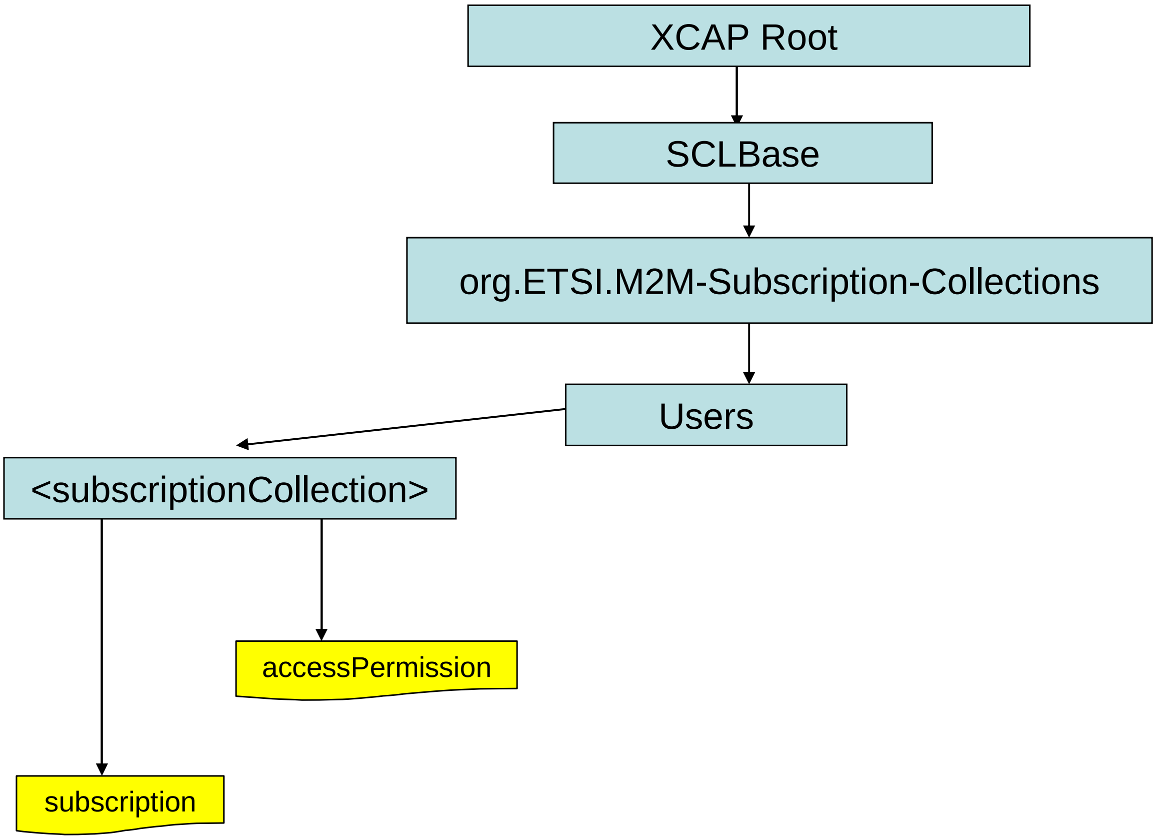


Figure F.25: Tree Structure for Subscription Collection Application Usage

F.1.2.20.2 Mapping between the XDMS XCAP <subscriptions> resource URL and M2M <subscriptions> resources

There are multiple locations within the M2M Resource tree structure where <subscriptions> resources are defined. As such, there multiple cases for mapping between XDMS XCAP <subscriptions> and M2M <subscriptions> resources. However, only one case shall be considered here.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.61: Subscription Collection M2M Resource URL Mapping <-> XCAP

M2M <subscriptions> Resources	XDMS XCAP <subscriptions> Resources
<sclBase>/scls/scl/subscriptions/subscription	XCAPbase/<XUI>/subscription
M2M NSCL shall set the proper permissions	XCAPbase/<XUI>accessPermission

F.1.2.20.3 Information Mapping Between XDMS XCAP <subscriptions> resource and M2M <subscriptions> resource

Table F.62: Subscription Collection XDMS Resource <-> M2M Resource

XDMS XCAP <groups> Resources	M2M <groups> Resources
XCAPbase/XUI/subscription	XDM document includes XCAP references to one or more subscription document stored under the XUI tree of the Subscription AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users<subsription>. The <subscription> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no subscription is defined in this collection.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.2.21 Management of Subscription resources

F.1.2.21.1 Subscription Application Usage

The M2M subscription Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read M2M subscription resources.

Every M2M subscription resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M subscription resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Subscriptions".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Subscriptions".

XML Schema

The M2M subscription XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in subscription.xsd, with the root element the sclBase, with the exception of child references.

- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.64.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M subscription resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents:

- The well-known name of the main M2M subscription resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The last well-known name of the M2M subscription resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

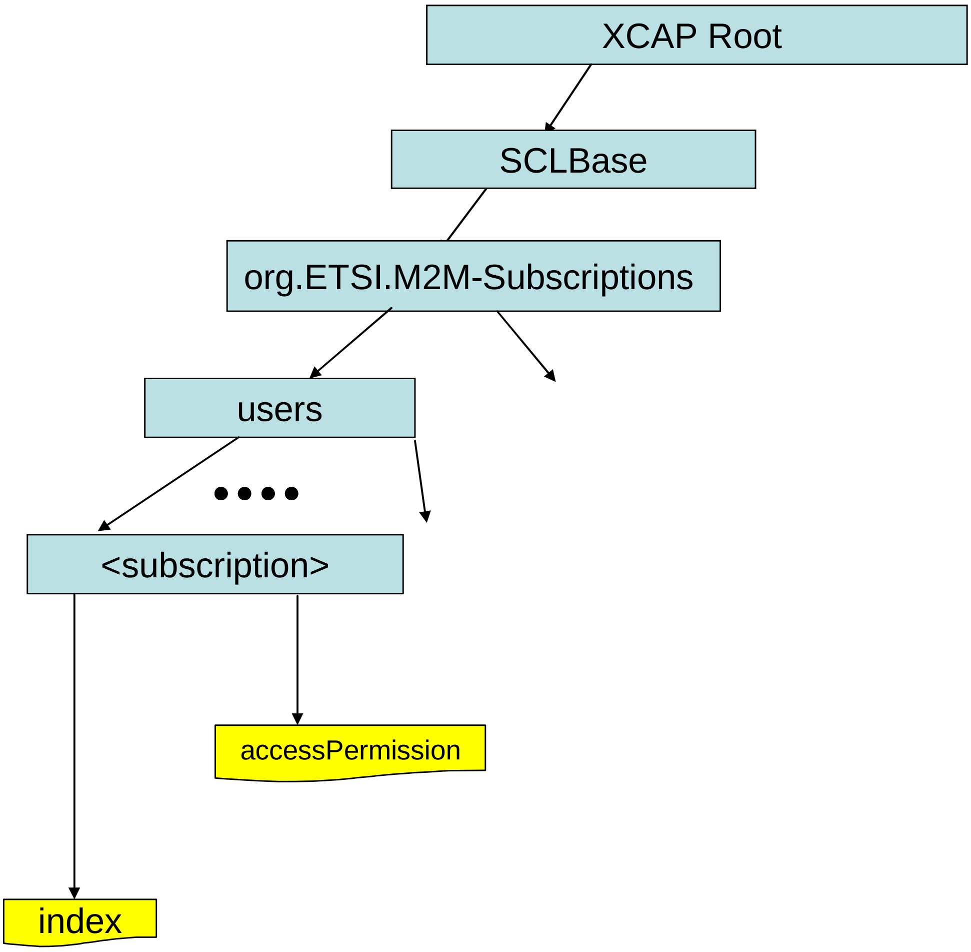


Figure F.26: Tree Structure for Subscription Application Usage

F.1.2.21.2 Mapping between the XDMS XCAP <subscription> resources and M2M <subscription> resources

There are multiple locations within the M2M Resource tree structure where <subscription> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <subscription> and M2M <subscription> resources. However, only one case shall be considered here.

It is to be noted that a 1:1 mapping exists between M2M < subscription> resources and XDMS XCAP < subscription> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.63: Subscription M2M Resource URL Mapping <-> XCAP URL

M2M < subscription> Resources	XDMS XCAP <subscription> Resources
<sclBase>/scls/<scl>/groups/subscriptions/<subscription>/attribute	XCAPbase/XUI/index
NSCL shall set the proper permissions	XCAPbase/XUI/accessPermission

F.1.2.21.3 Information Mapping Between XDMS XCAP <subscription> resources and M2M <subscription> resources

Table F.64: Subscription XDMS Resource <-> M2M Resource

XDMS XCAP <subscription> Resource	M2M <subscription> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <subscription> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.2.22 Management of m2mPocs Collection resources

F.1.2.22.1 m2mPoc Collection Application Usage

The m2mPoc Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of m2mPoc resources.

Each m2mPoc Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the m2mPoc Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-M2mPoc-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-m2mPoc-Collections".

XML Schema

The m2mPoc Collections Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in m2mPocs.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- M2mPoc document, per XUI, under users' tree shall conform to the XML schema as defined in table F.66.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.66.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an m2mPoc Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be three well-known m2mPoc Collections XDM documents. They are as follows:

- The main well-known name of the m2m2Poc Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/ index".
- The second well-known name of the m2mPoc Collection resource XDM document shall be "m2mPoc". The document selector to access the "m2mPoc" XDM document shall be "[auid]/users/[xui]/ m2mPoc".
- The third well-known name of the accessPermission Collection resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/ accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

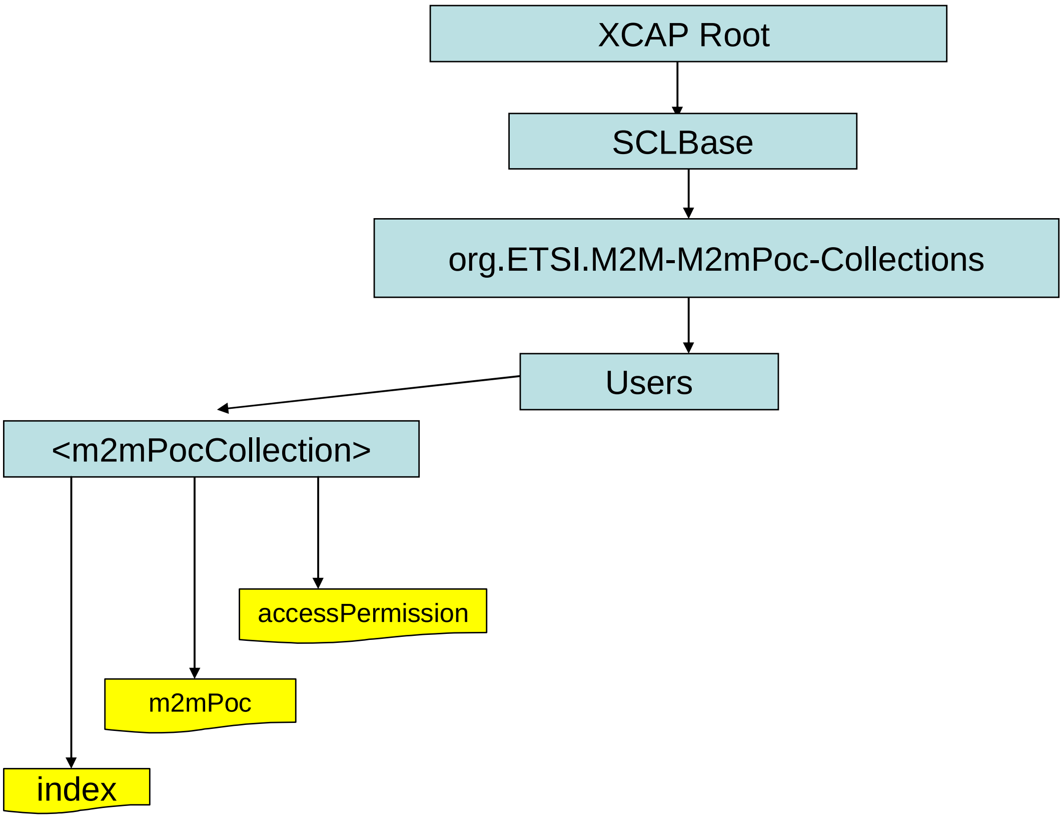


Figure F.27: Tree Structure for m2mPoc Collection Application Usage

F.1.2.22.2 Mapping between the XDMS XCAP <m2mPocs> resource URL and M2M <m2mPocs> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.65: m2mPoc Collection M2M Resource URL Mapping <-> XCAP

M2M <m2mPocs> Resources	XDMS XCAP <m2mPocs> Resources
<sclBase>/scls/<scl>/ m2mPocs/attribute	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/ m2mPocs/m2mPoc	XCAPbaseXUI/m2mPoc
The M2M NSCL sets it in such a way to ensure that only the registered SCL can perform permissible operations.	XCAPbase/XUI/accessPermission

F.1.2.22.3 Information Mapping Between XDMS XCAP <m2mPocs> resource and M2M <m2mPocs> resource

Table F.66: m2mPoc Collection XDMS Resource <-> M2M Resource

XDMS XCAP <m2mPocs> Resources	M2M <m2mPocs> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <m2mPocs> collection
XCAPbase/XUI/m2mPoc	XDM document includes XCAP references to one or more m2mPoc document stored under the XUI tree of the m2mPoc AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users<m2mPoc>. The <m2mPoc> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no m2mPoc is defined in this collection
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.1.2.23 Management of m2mPoc resources

F.1.2.23.1 m2mPoc Application Usage

The M2M m2m2Poc Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read M2M m2mPoc resources.

Every M2M m2mPoc resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M m2mPoc resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-M2mPocs".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-mMm2Pocs".

XML Schema

The M2M m2mPoc XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in m2mPoc.xsd, with the root element the sclBase, with the exception of child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.68.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M m2mPoc resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents:

- The well-known name of the main M2M m2mPoc resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the M2M m2mPoc resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

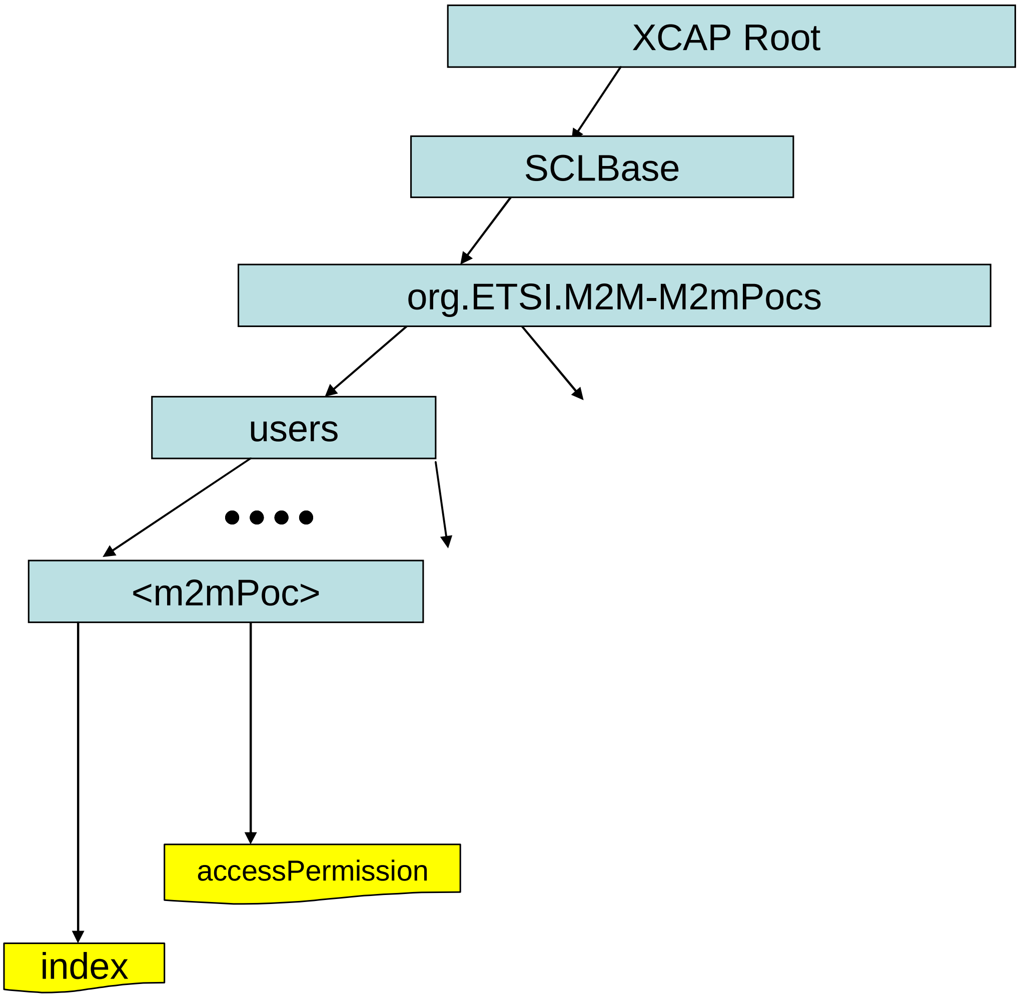


Figure F.28: Tree Structure for m2mPoc Application Usage

F.1.2.23.2 Mapping between the XDMS XCAP <m2mPoc> resources and M2M <m2mPoc > resources

It is to be noted that a 1:1 mapping exists between M2M <m2mPoc> resources and XDMS XCAP <m2mPoc> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-M2m2pocs/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.67: m2mPoc M2M Resource URL Mapping <-> XCAP URL

M2M <m2mPoc> Resources	XDMS XCAP <m2mPoc> Resources
<sclBase>/scls/<scl>/m2mPocs/<m2mPoc>/attribute	XCAPbase/XUI/index
The NSCL shall ensure that the proper access rights are set to ensure that only registering SCLs can perform the required operations	XCAPbase/XUI/accessPermission

F.1.2.23.3 Information Mapping Between XDMS XCAP <m2mPoc> resources and M2M <m2m2Poc> resources

Table F.68: m2mPoc XDMS Resource <-> M2M Resource

XDMS XCAP <m2mPoc> Resource	M2M <m2mPoc> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the < m2mPoc > resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.2.24 Management of NotificationChannel Collection resources

F.1.2.24.1 NotificationChannel Collection Application Usage

The NotificationChannel Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of notificationChannel resources.

Each notificationChannel Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the notificationChannel Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-NotificationChannel-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-NotificationChannel-Collections".

XML Schema

The notificationChannel Collections Resource XDM documents shall conform to the following XML schemas:

- NotficationChannel document, per XUI, under users' tree shall conform to XML schema as defined in table F.70.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.70.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a notificationChannel Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known notificationChannel Collections XDM documents. They are as follows:

- The first well-known name of the notificationChannel Collection resource XDM document shall be "notificationChannel". The document selector to access the "notificationChannel" XDM document shall be "[auid]/users/[xui]/ notificationChannel".
- The well-known name of the accessPermission resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

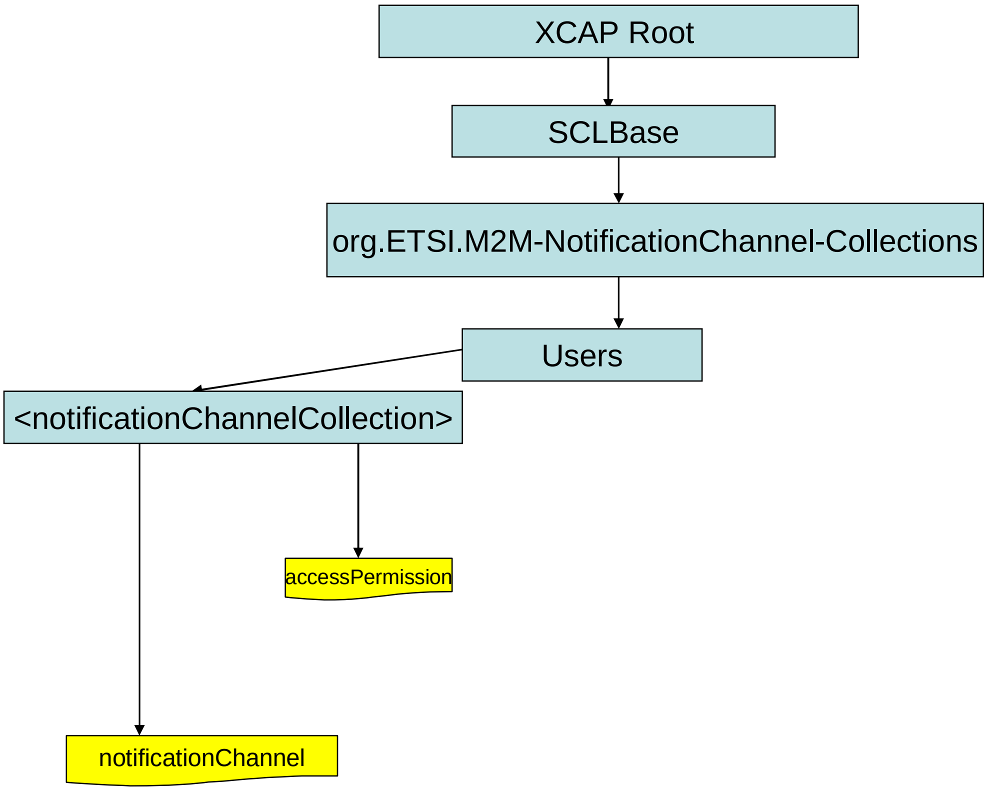

XCAP Root
SCLBase
org.ETSI.M2M-NotificationChannel-Collections
Users
<notificationChannelCollection>
accessPermission
notificationChannel

Figure F.29: Tree Structure for NotificationChannel Collection Application Usage

F.1.2.24.2 Mapping between the XDMS XCAP <notificationChannels> resource URL and M2M <notificationChannels> resources

There are multiple locations within the M2M Resource tree structure where <notificationChannels> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <notificationChannels> and M2M <notificationChannels> resources. However, only one case shall be described here.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.69: notificationChannel Collection M2M Resource URL Mapping <-> XCAP

M2M <notificationChannels> Resources	XDMS XCAP <notificationCannels> Resources
<sclBase>/scls/<scl>/ notificationChannels>/<notificationChannel>	XCAPbase/<XUI>/notificationChannel
The NSCL has to set it to allow all access.	XCAPbase/<XUI>/accessPermission

F.1.2.24.3 Information Mapping Between XDMS XCAP <notificationChannels> resource and M2M <notificationChannels> resource

Table F.70: Notification Channel Collection XDMS Resource <-> M2M Resource

XDMS XCAP <notificationChannels> Resources	M2M < notificationChannels > Resources
XCAPbase/XUI/notificationChannel	XDM document includes XCAP references to one or more notificationChannel document stored under the XUI tree of the Notification Channel AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users<notificationChannel>. The <notificationChannel> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no notificationChannel is defined in this collection.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.2.25 Management of NotificationChannel resources

The M2M notificationChannel Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read M2M notificationChannel resources.

Every M2M notificationChannel resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M notificationChannel resources.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Notification-Channels".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Notification-Channels".

XML Schema

The M2M notificationChannel XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in notificationChannels.xsd, with the root element the sclBase, with the exception of child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.72.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M notificationChannel resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be two well-known documents:

- The well-known name of the main M2M notificationChannel resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The well-known name of the second M2M notificationChannel resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

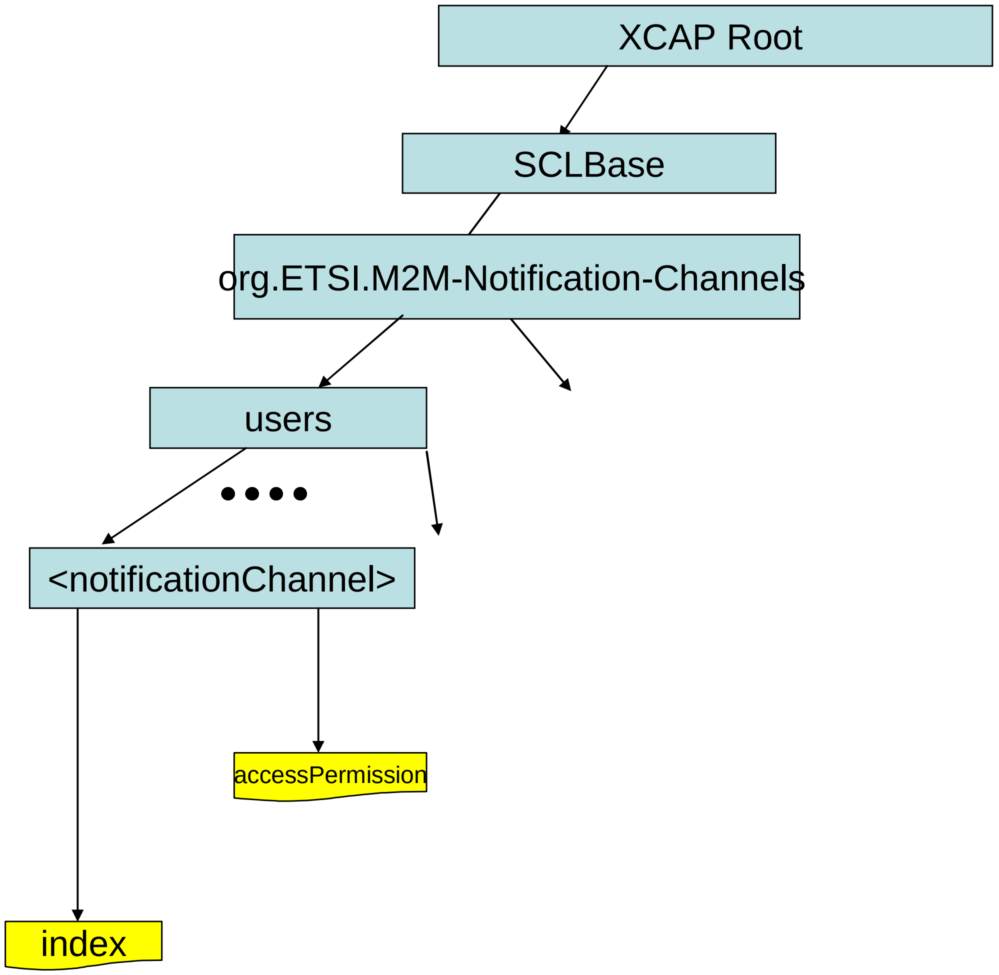


Figure F.30: Tree Structure for Notification Channels Application Usage

F.1.2.25.1 Mapping between the XDMS XCAP <notificationChannel> resources and M2M <notificationChannel> resources

There are multiple locations within the M2M Resource tree structure where <notificationChannels> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <notificationChannels> and M2M <notificationChannel> resources. However, only one case shall be described here.

It is to be noted that a 1:1 mapping exists between M2M <notificationChannel> resources and XDMS XCAP < notificationChannel> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-NotifiationChannels/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.71: notificationChannel M2M Resource URL Mapping <-> XCAP URL

M2M <notificationChannel> Resources	XDMS XCAP <notificationChannel> Resources
<sclBase>/scls/<scl>/notificationChannels/<notificationChannel>/attribute	XCAPbase/XUI/index
Not applicable. The M2M NSCL has to set it to allow all access	XCAPbase/XUI/accessPermission

F.1.2.25.2 Information Mapping Between XDMS XCAP <notificationChannel> resources and M2M <notificationChannel> resources

Table F.72: notificationChannel XDMS Resource <-> M2M Resource

XDMS XCAP <notificationChannel> Resource	M2M <notificationChannel> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <notification> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.1.3 M2M to XDMS URI Mapping Principles

F.1.3.1 Fetching information from XCAP

The NSCL has the task of fetching information from XDMS in response to an incoming M2M request. To that effect the following steps shall be undertaken:

1) The information to be read from the XDMS by the M2M NSCL depends on the incoming request. The following cases, described later in more details, can be distinguished:

   - TargetID identifies a collection.
   - TargetID identifies a named resource.
   - TargetID identifies an element in a named resource or a sub-element in a named resource commonly referred to as partial addressing, for example a specific attribute (except collection sub-resources).
   - TargetID identifies an accessRighID.

2) For reading information from XDMS, the NSCL shall translate a targetID or any M2M URI derived by the NSCL to an XCAP URI. Clause F.1.3.1.1 covers that.

3) Where applicable, and for XCAP URIs that have to be returned (a reference to group, container, etc.) to a request originator procedure (clause F.1.3.1.4) have to be used for the purpose of translating the XCAP URI to an M2M URI.

TargetID

Within the context of this section and remaining sections, TargetID includes the resource URI plus the data within the addressed resource, if applicable. This data can include a sub-resource, a specific attribute, or subElements within an attribute.

TargetID identifies a collection (an entire collection)

In this case the NSCL shall return all elements in the collection. These are stored in XDMS as XCAP URIs and shall be converted to appropriate M2M URI before they can be returned to the request originator. Reading collections require that the application usage that corresponds to the requested collection be used instead of the application usage that corresponds to the targetID. The target XDM document to be read from the collection, by the NSCL, in this case shall be derived from the identified collection application usage that corresponds to the incoming targetID (see clause G.2.2 for additional information as well as the example below). The identified target XDM document to be read by the NSCL, from XDMS, as per step 2 above includes a list of XCAP URIs that have to be returned to the request originator. They have to be translated as per step 3 (above) before they can be returned to the request originator.

TargetID identifies a named resource (entire resource)

In this case, the NSCL shall read all attributes, where defined, and shall return references for all sub-resources under the named resource. The attributes shall be read from the XDMS. The application usage that corresponds to the targetID defines the XDM documents that shall be read by the NSCL for that purpose.

To return references to sub-resources, the NSCL shall not read anything from XDMS. Rather the XDM document names, identified from the application usage that corresponds to the targetID, depicting the references, shall be appended to the targetID and returned. In this case, no additional translation is required. This is repeated for every reference to be returned.

For the attributes, the element attribute is appended to the targetID followed by the actual attribute information. This is repeated for every attribute to be returned.

TargetID identifies an attribute or subElement commonly referred to as partial addressing in a named resource, for example a specific attribute such as an accessRightID in a named resource

This implies that the NSCL shall identify the XDM document to be read from XDMS and that corresponds to the required element to be read. The application usage that corresponds to the targetID and the tables therein shall be used for that purpose. Once the XDM document is identified, it is read by the NSCL and the required information is returned in a response to the request originator.

If the read information from XDMS includes an XCAP URI, this reference shall be translated using step 3 above before it can be returned back to the request originator\.

TargetID identifies an accessRightID

In this case, the NSCL shall read the accessPermission document for the application usage that corresponds to the incoming targetID. The NSCL shall than locate the internal URI that corresponds to the XCAP URI included in the XDM document and shall return that reference to the request originator, No translation shall be performed in this case (see clause G.2.1.4 for additional information on the internal URI).

F.1.3.1.1 Mapping Between TargetID and XDMS Resource URI

There is a one to one mapping between every XCAP URI used in XDMS and TargetID used over the mId, dIa, and mIa interfaces. The mapping shall performed by the M2M NSCL.

Initially the XCAP root shall be provisioned in the M2M NSCL or discovered through defined XDMS procedures.

Following that, and depending on the application usage, the M2M NSCL shall create the necessary XCAP URI that corresponds to the targetID. The XCAP URI has 3 components as follows:

- XCAP URI = = XCAPBase/XUI/ XDM document.

If the targetID identifies the SCLBase Application Usage then the XCAP URI is constructed from table F.2. The entry in the table that matches the targetID is the selected XCAP URI.

For all other application usages, the various components are constructed as follows:

- XCAP base depends on the application usage derived from the targetID. In case of collections, the XCAP base corresponds to the identified collection Application Usage derived from the targetID as explained above.
- XUI which is determined as defined in clause F.1.3.1.3.
- Actual target XDM document. This is derived by locating XDM documents corresponding to the elements following the named element in the targetID depending on the application usage. If there are no elements following the named element in the targetID, than the M2M NSCL shall use the application usage that corresponds to the targetID to derive the appropriate XDM documents to be read. There are cases where more than one XDM document have to be read by the NSCL depending on the application usage that corresponds to the targetID (see above section entitled **"TargetID identifies a named resource (entire resource)").**

F.1.3.1.2 Application Usage Base XCAP URI

Every application usage shall have an XCAP base XCAP URI allocated to it. This includes the root XCAP UR and including the /users/ element or /global/element. Each application usage shall identify its base XCAP URI.

F.1.3.1.3 Deriving XUI

For the purpose of using XDMS, the addressed resource in the targetID over the dIa or mIa interface shall be mapped to an XDMS XUI by the M2M NSCL at resource creation and storage in XDMS.

This requires first that all paths segments after the named resource are removed before the mapping is performed. Note that the targetID for the SCLBase application usage has only /sclBase/ as the targetID.

The mapping shall be achieved through a translation function that works both directions. Hence, the function has the following properties:

- F (modified TargetID) = XUI.
- F^{-1} (XUI) = modified TargetID.

Where F^{-1} is the inverse of F.

This function is implementation specific. A simple and straight forward example of such a function is a function that uses URI encoding of modified targetID to derive XUI.

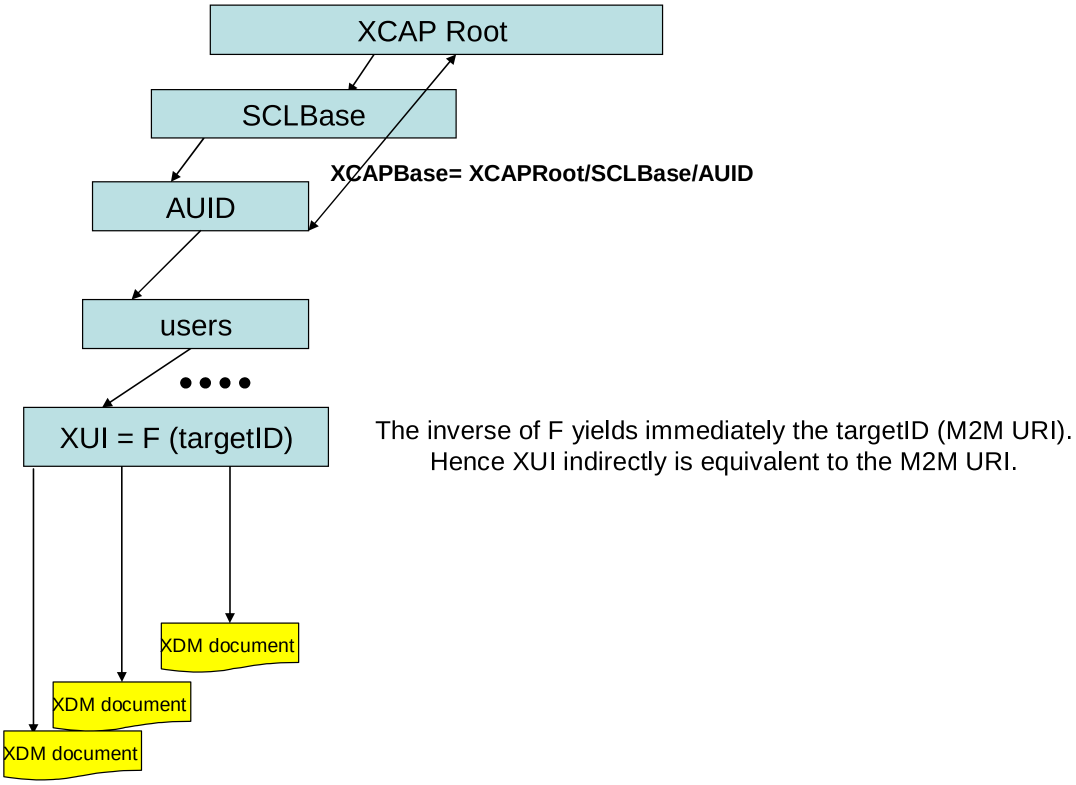

XCAP Root
SCLBase
XCAPBase= XCAPRoot/SCLBase/AUID
AUID
users
XUI = F (targetID)
The inverse of F yields immediately the targetID (M2M URI).
Hence XUI indirectly is equivalent to the M2M URI.
XDM document
XDM document
XDM document

Figure F.31: Translation Function (deriving XUI)

F.1.3.1.4 XCAP URI to M2M URI Mapping

If the XCAP URI identifies a reference that shall be returned in a response, it shall be first converted into an M2M URI. Any XCAP URI has the XUI as the last element in the URI.

The complete M2M URI is derived by applying the following translation on the XUI element in the XCAP URI:

- F^{-1} (XUI) = M2M URI.

This is indeed the inverse of the translation function that created the XUI.

Figure F.32 illustrates the above with an example for how to return a list of announced applications within an application collection that belongs to a registered SCL.

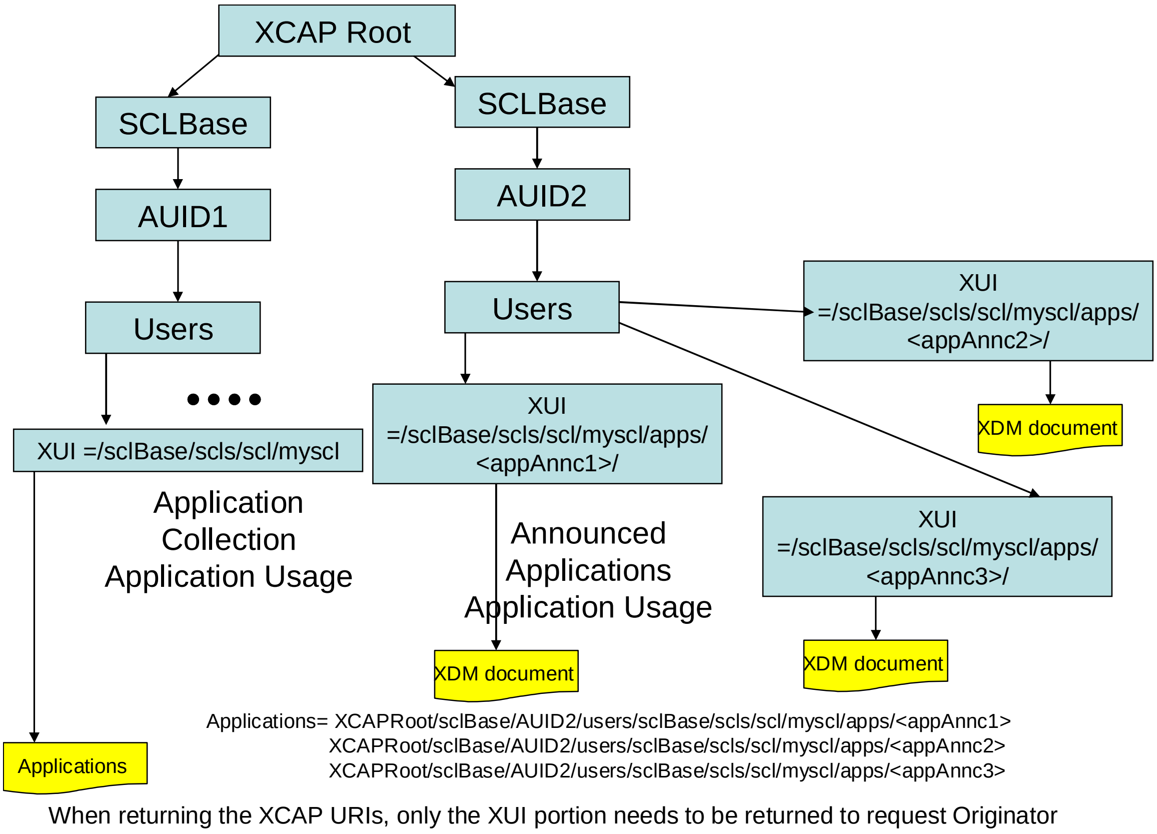

XCAP Root
SCLBase
AUID1
Users
••••
XUI =/sclBase/scls/scl/myscl
Application Collection Application Usage
Applications
SCLBase
AUID2
Users
XUI =/sclBase/scls/scl/myscl/apps/ <appAnnc1>/
Announced Applications Application Usage
XDM document
XUI =/sclBase/scls/scl/myscl/apps/ <appAnnc2>/
XDM document
XUI =/sclBase/scls/scl/myscl/apps/ <appAnnc3>/
XDM document
Applications= XCAPRoot/sclBase/AUID2/users/sclBase/scls/scl/myscl/apps/<appAnnc1>
XCAPRoot/sclBase/AUID2/users/sclBase/scls/scl/myscl/apps/<appAnnc2>
XCAPRoot/sclBase/AUID2/users/sclBase/scls/scl/myscl/apps/<appAnnc3>
When returning the XCAP URIs, only the XUI portion needs to be returned to request Originator

Figure F.32: XCAP to M2M URI Mapping (Example)

Illustrative example to retrieve the list of groups in a group collection in a registered SCL

GET http://myoperator.org/scls/myScl/groups (HTTP request on M2M level to retrieve all groups in the group collection in an scl identified as myScl).

- 200 OK.

The response would look as follows:

```
<groups>
<namedRef id="test">http://myoperator.org/scls/myScl/groups/firstgroup<namedRef>
<namedRef id="yetAnotherGroup">http://myoperator.org/scls/myScl/groups/secondgroup<namedRef>
</groups>
```

The first returned group ID <firstgroup> is derived as follows.

Based on the targetID, the application usage is registered SCL Application Usage, and from this application usage the path "groups" identifies the group Collection. Hence the group collection application usage corresponds to the targetID.

The XCAP base for the group collection Application Usage shall be selected.

The XUI is then derived based on the targetID stripping the "groups" element since it is after the named resource.

Now the XDM document to be read in the Group collection Application Usage is the group document. Hence, the list of XCAP references in the Group collection application usage are read from the group XDM document. In this list, the first element had an XCAP URI where the XUI in there was, for example equivalent, to XUI-Group1. Applying the inverse of the translation function below yields the M2M URI.

- F^{-1} (XUI-Group1) = http://myoperator.org/scls/myScl/groups/firstgroup.

This will be then the first element to be returned.

The same algorithm applies to the second XCAP URI which will not be repeated here.

F.1.3.2 XUI translation to SIP URL

To comply to XDMS requirements on XUI syntax, the M2M NSCL shall translate the chosen XDMS XUI to a SIP URL where the XUI translates to the user part portion of the SIP URL. The M2M NSCL shall perform the reverse operation for information retrieved from the XDMS when needed.

F.2 NSCL Procedures

This clause provides a list of the basic procedures that shall be performed by M2M NSCL to manage the different M2M resources stored in XDMS.

It is first assumed that the M2M NSCL is aware that the XDMS supports all the application usages required for M2M as per clause F.1.1. Optionally the M2M NSCL implements the discovery procedure defined in RFC 4825 [i.4] for XCAP server capabilities to ensure that the XDMS supports all application usages required for M2M as per clause F.1.

F.2.1 Resource Creation

The following sections present typical examples for various call flows for creating the various M2M resources for the purpose of storing information regarding these resources in XMDS. The following are some of the generic guiding principles:

- The XUI created for a collection is identical to the XUI for its parent (see figure F.33 for an example for an application collection with a registered SCL).
- All collections are created at the time the parent resource is created even if they are initially empty.

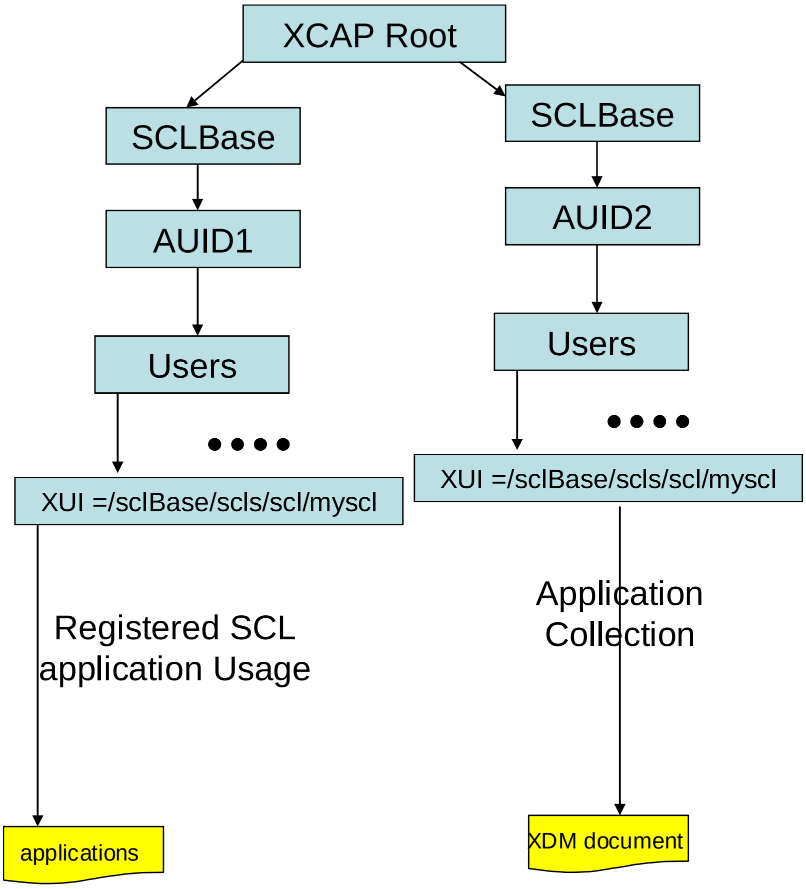


Figure F.33: Collections and Parent Relationship for XUI

F.2.1.1 Creation of an scl Resource

Figure F.34 show the procedure to be performed by the NSCL to create a registered SCL XDM resource. All collections and sub-resources are created even if they are empty.

The Registered SCL Application Usage applies in the case an SCL resource is created as a result of a successful SCL registration. Every registered SCL resource is a member of an SCL collection resource (there is only one SCL collection resource hence by default all registered SCLs are implicitly in the SCL collection). For a newly registered SCL, the NSCL shall create a new tree representing the SCL resource under the users' tree for the Registered SCL Application Usage. The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUI/) for the registered SCL resource applying procedure (clause F.1.3.1) in that regard.

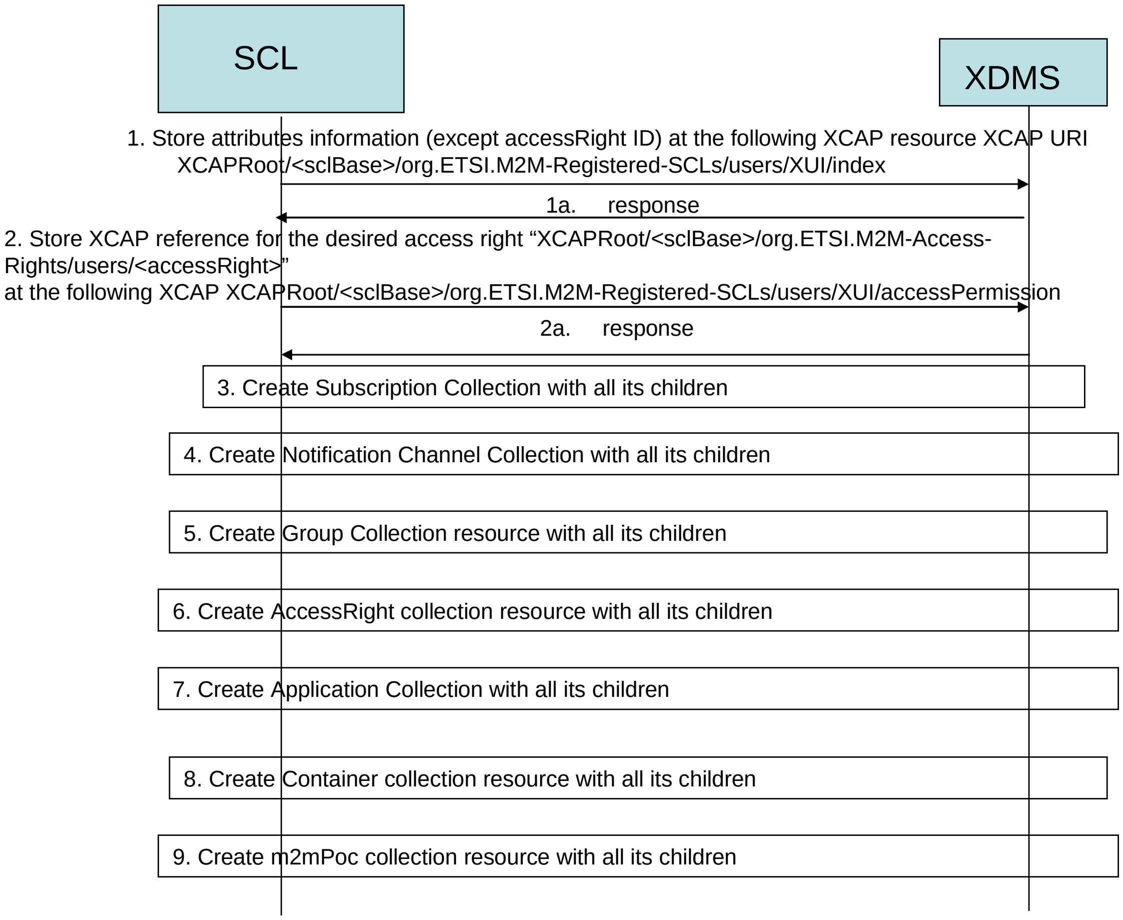

SCL
XDMS
1. Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right “XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>”
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUI/accessPermission
2a. response
3. Create Subscription Collection with all its children
4. Create Notification Channel Collection with all its children
5. Create Group Collection resource with all its children
6. Create AccessRight collection resource with all its children
7. Create Application Collection with all its children
8. Create Container collection resource with all its children
9. Create m2mPoc collection resource with all its children

Figure F.34: Registered SCL Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the registered SCL in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs /users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP URI resource for it.

- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- In step 3, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 4, NSCL creates a Notification Channel collection resource using procedure (clause F.2.1.18) for that matter.

- In step 5, NSCL creates a Group collection resource using procedure (clause F.2.1.12) for that matter.

- In step 6, NSCL creates an accessRight collection resource using procedure (clause F.2.1.14) for that matter.

- In step 7, NSCL creates an Application Collection resource using procedure (clause F.2.1.15) for that matter.

- In step 8, NSCL creates a container collection resource using procedure (clause F.2.1.13) for that matter.

- In step 9, NSCL creates an m2mPoC collection resource using procedure (clause F.2.1.17) for that matter.

F.2.1.2 Creation of Announced Application Resource

Figures F.35 show the procedure to be performed by the NSCL to create an Announced Application XDM resource.

The Announced Application Application Usage applies in the case an announced application resource is added as member of an application collection resource that belongs to a registered SCL.

In this case, the NSCL shall create a new tree representing the announced application resource under the users' tree for the Announced Application "Application Usage". The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI/) for the announced application applying procedure (clause F.1.3.1) in that regard.

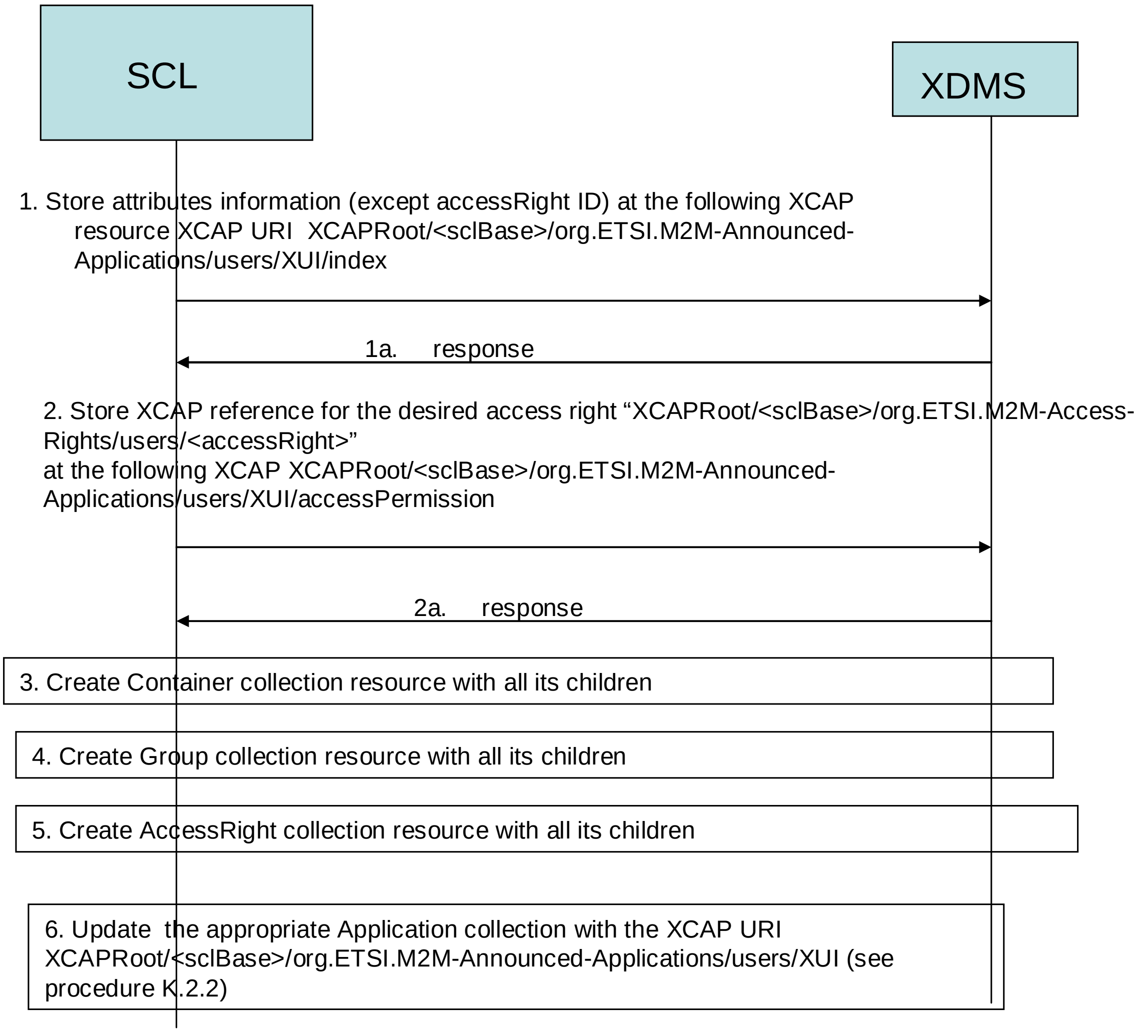

SCL
XDMS
1. Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI/accessPermission
2a. response
3. Create Container collection resource with all its children
4. Create Group collection resource with all its children
5. Create AccessRight collection resource with all its children
6. Update the appropriate Application collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI (see procedure K.2.2)

Figure F.35: Announced Application Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the applicationAnnc in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Applications/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- In step 3, NSCL creates a container collection resource using procedure (clause F.2.1.13) for that matter.
- In step 4, NSCL creates a Group collection resource using procedure (clause F.2.1.12) for that matter.
- In step 5, NSCL creates an accessRight collection resource using procedure (clause F.2.1.14) for that matter.
- The NSCL updates the appropriate Application Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.3 Creation of Local application Resource

Figures F.36 shows the procedure to be performed by the NSCL to create a Local application XDM resource.

The SCL Local application Application Usage applies in the case a newly local application resource successfully registers and becomes a member of an application collection resource that belongs to the SCLBase

In this case, the NSCL shall create a new tree representing the local application resource under the users' tree for the Local application Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI/) for the local application resource applying procedure (clause F.1.3.1) in that regard.

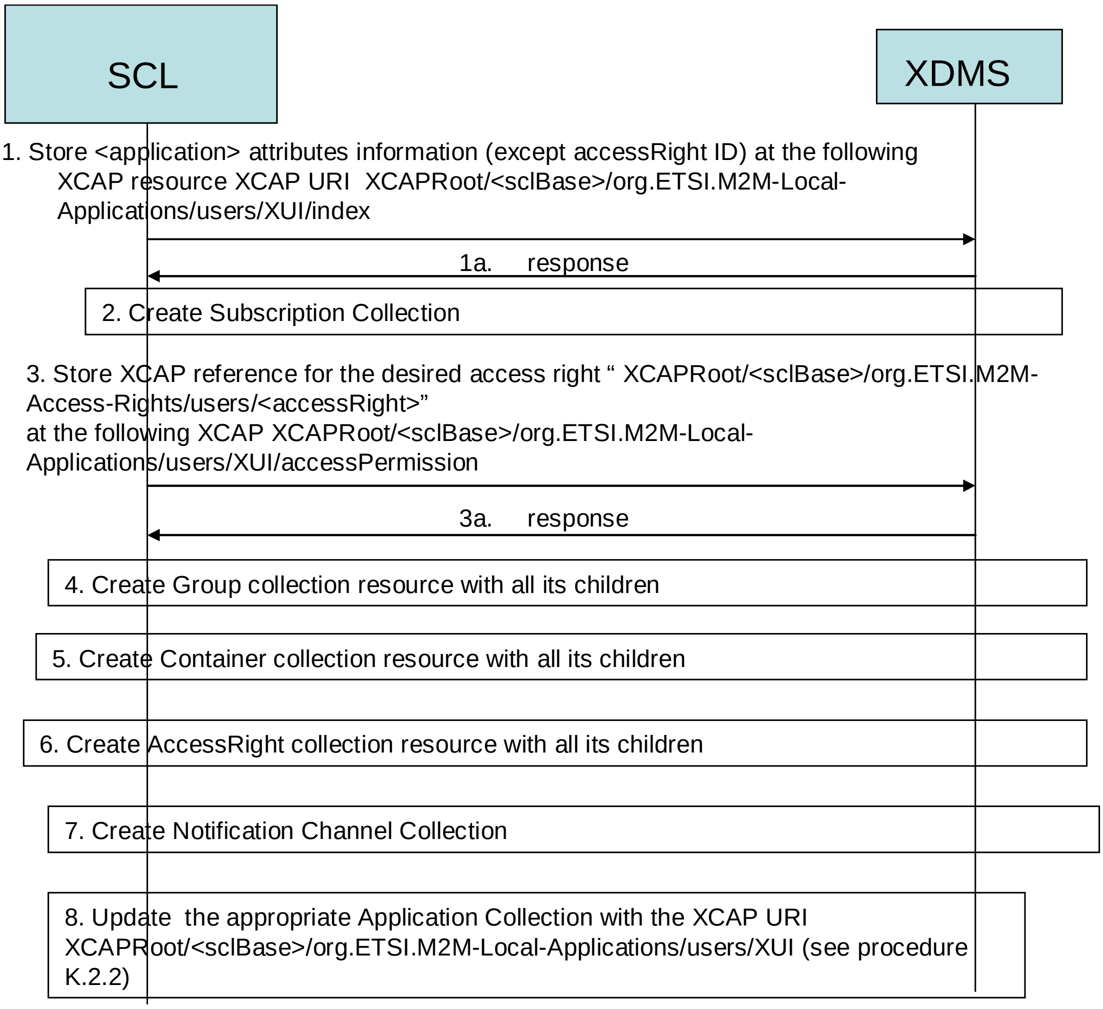

SCL
XDMS
1. Store <application> attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI/accessPermission
3a. response
4. Create Group collection resource with all its children
5. Create Container collection resource with all its children
6. Create AccessRight collection resource with all its children
7. Create Notification Channel Collection
8. Update the appropriate Application Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI (see procedure K.2.2)

Figure F.36: Local Application Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the local application in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Local-Applications/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, NSCL creates a container collection resources using procedure (clause F.2.1.13) for that matter.
- In step 5, NSCL creates a Group collection resource using procedure (clause F.2.1.12) for that matter.
- In step 6, NSCL creates an accessRight collection resource using procedure (clause F.2.1.14) for that matter.
- In step 7, NSCL creates a Notification Channel collection resource using procedure (clause F.2.1.18) for that matter.
- The NSCL updates the appropriate Application Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.4 Creation of AccessRight Resource

Figure F.37 shows the procedure to be performed by the NSCL to create an AccessRight XDM resource.

The AccessRight Application Usage applies in the case a newly created accessRight resource is added as a member of an accessRight collection resource that belongs to any of the following M2M resources:

- Case 1: The accessRight collection resource belongs to a registered SCL.
- Case 2: The accessRight collection resource belongs to the sclBase.
- Case 3: The accessRight collection resource belongs to a registered local application.
- Case 4: The accessRight collection resource belongs to an announced application.

For all the above cases, the NSCL shall create a new tree representing the accessRight resource under the users' tree for the AccessRight Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI/) for the access right resource applying procedure (clause F.1.3.1) in that regard.

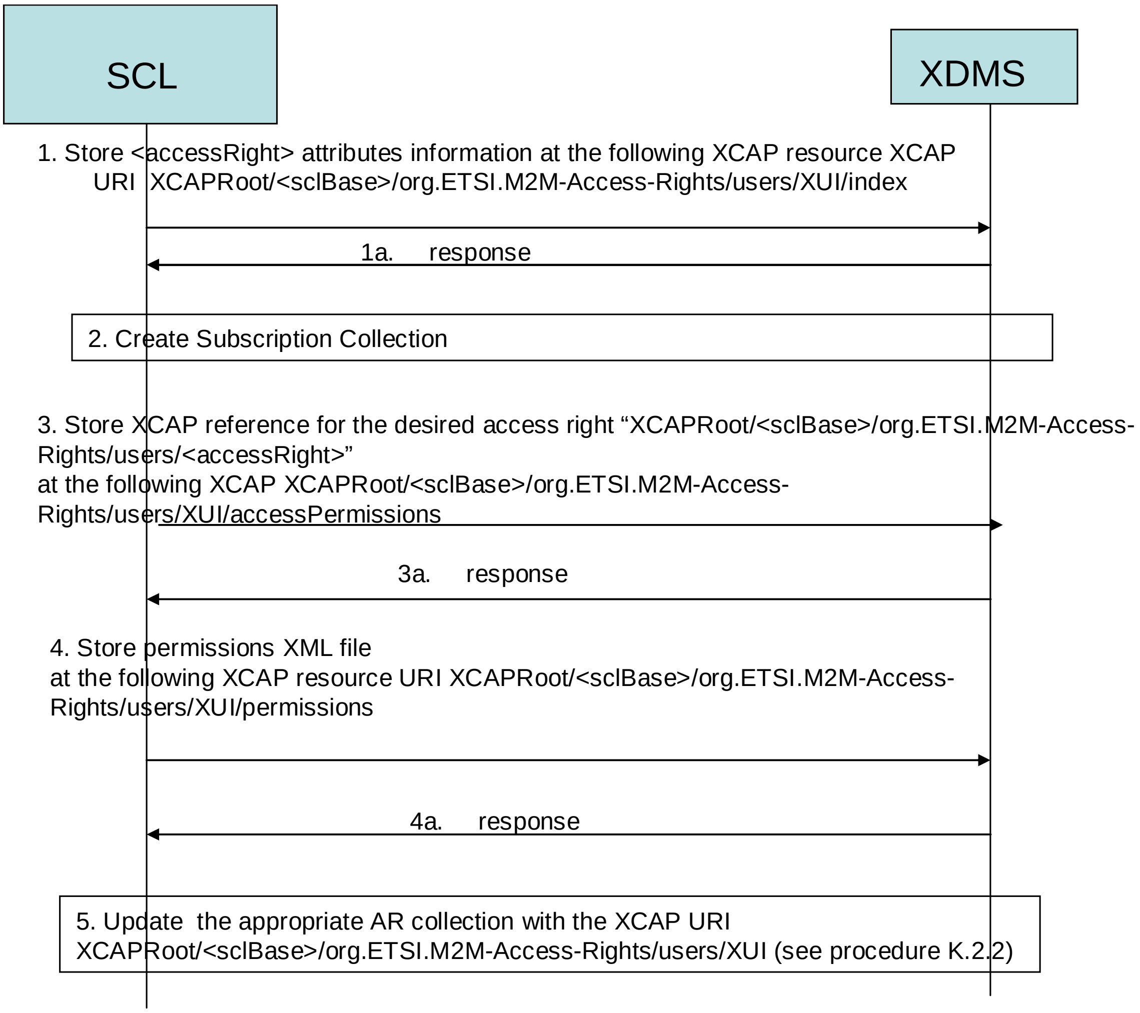

SCL
XDMS
1. Store <accessRight> attributes information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI/accessPermissions
3a. response
4. Store permissions XML file
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI/permissions
4a. response
5. Update the appropriate AR collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI (see procedure K.2.2)

Figure F.37: Access Right Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRights/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XML document that identifies the actual access permission associated with the right to modify permissions associated with the accessRight resource in the XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRights/users/XUI/accessPermission.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the NSCL stores the document that identifies the actual permissions associated with the accessRight resource in the XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRights/users/XUI/permissions.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate ARCollection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.4.1 Handling of Access Rights Resources

Access rights resources require special handling in the NSCL. Access rights resources are typically created and than allocated later to a specific M2M resource.

As such, when the NSCL receives a request to create an access right resource it shall create the access right resource using the above procedure. Following that, the NSCL shall maintain internally a mapping between the URI for the access right resource it returned to the request originator and the XCAP URI for the created access right resource in XDMS.

At a later time, when the access resource is allocated to an M2M URI resource, the NSCL shall fetch the XCAP URI that matches the requested access resource URI (in the incoming M2M request) for the M2M resource and shall then update the proper accessPermission XDM document in the proper application usage depending on the incoming request.

F.2.1.5 Creation of Group Resource

Figure F.38 shows the procedure to be performed by the NSCL to create a Group XDM resource.

The Group Application Usage applies in the case a newly created group resource is added as a member of a group collection resource that belongs to any of the following M2M resources:

- Case 1: The group collection resource belongs to a registered SCL.
- Case 2: The group collection resource belongs to the sclBase.
- Case 3: The group collection resource belongs to a registered local application.
- Case 4: The group collection resource belongs to an announced application.

For all the above cases, the NSCL shall create a new tree representing the group resource under the users' tree for the Group Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI/) for the group applying procedure (clause F.1.3.1) in that regard.

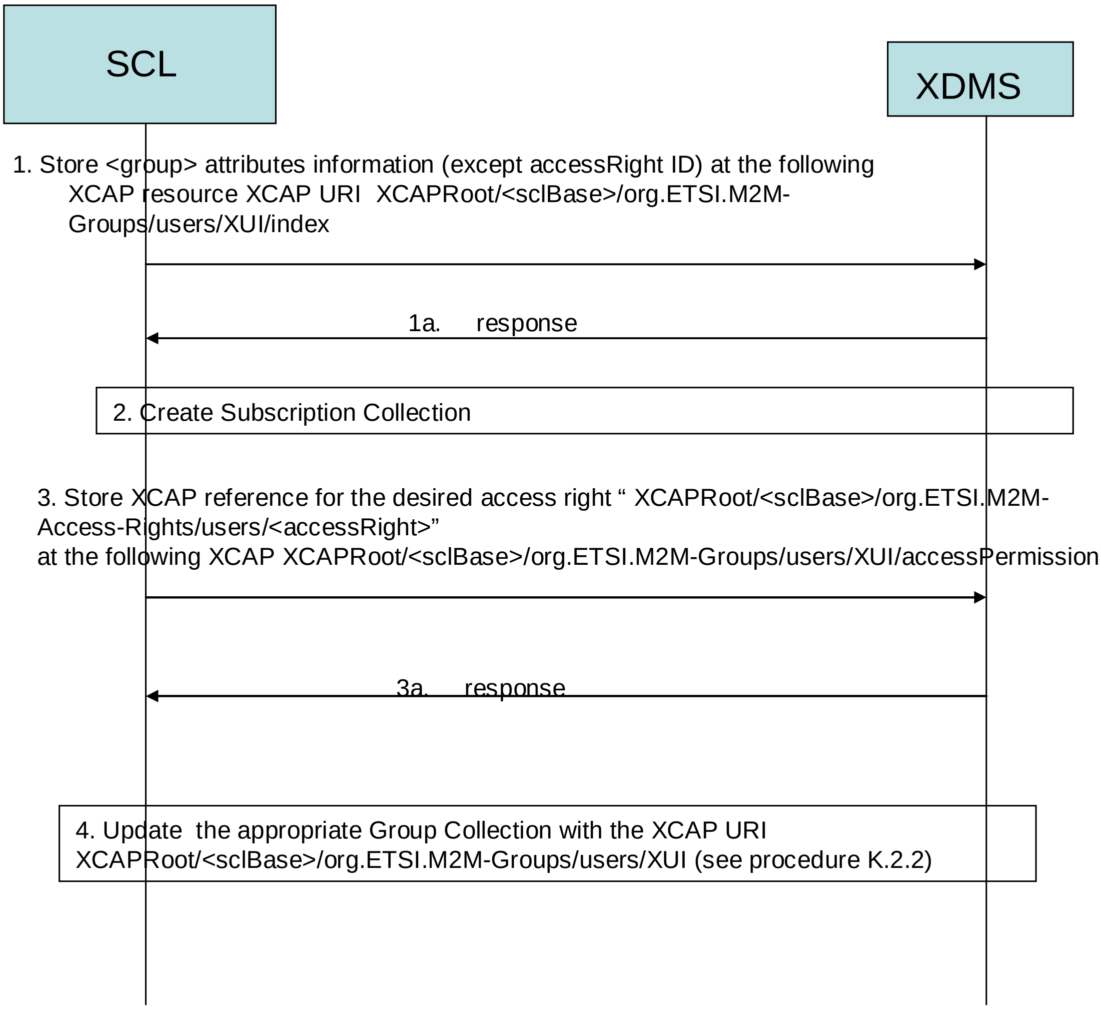

SCL
XDMS
1. Store <group> attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI/accessPermission
3a. response
4. Update the appropriate Group Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI (see procedure K.2.2)

Figure F.38: Group Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the group resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Groups/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate Group Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.6 Creation of Container Resource

Figure F.39 shows the procedure to be performed by the NSCL to create a Container XDM resource.

The Container Application Usage applies in the case a newly created container resource is added as a member of a container collection resource that belongs to any of the following M2M resources:

- Case 1: The container collection resource belongs to a registered SCL.

- Case 2: The container collection resource belongs to the sclBase.
- Case 3: The container collection resource belongs to a registered local application.
- Case 4: The container collection resource belongs to an announced application.

For all the above cases, the NSCL shall create a new tree representing the container resource under the users' tree for the Container Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Container/users/XUI/) for the container applying procedure (clause F.1.3.1) in that regard.

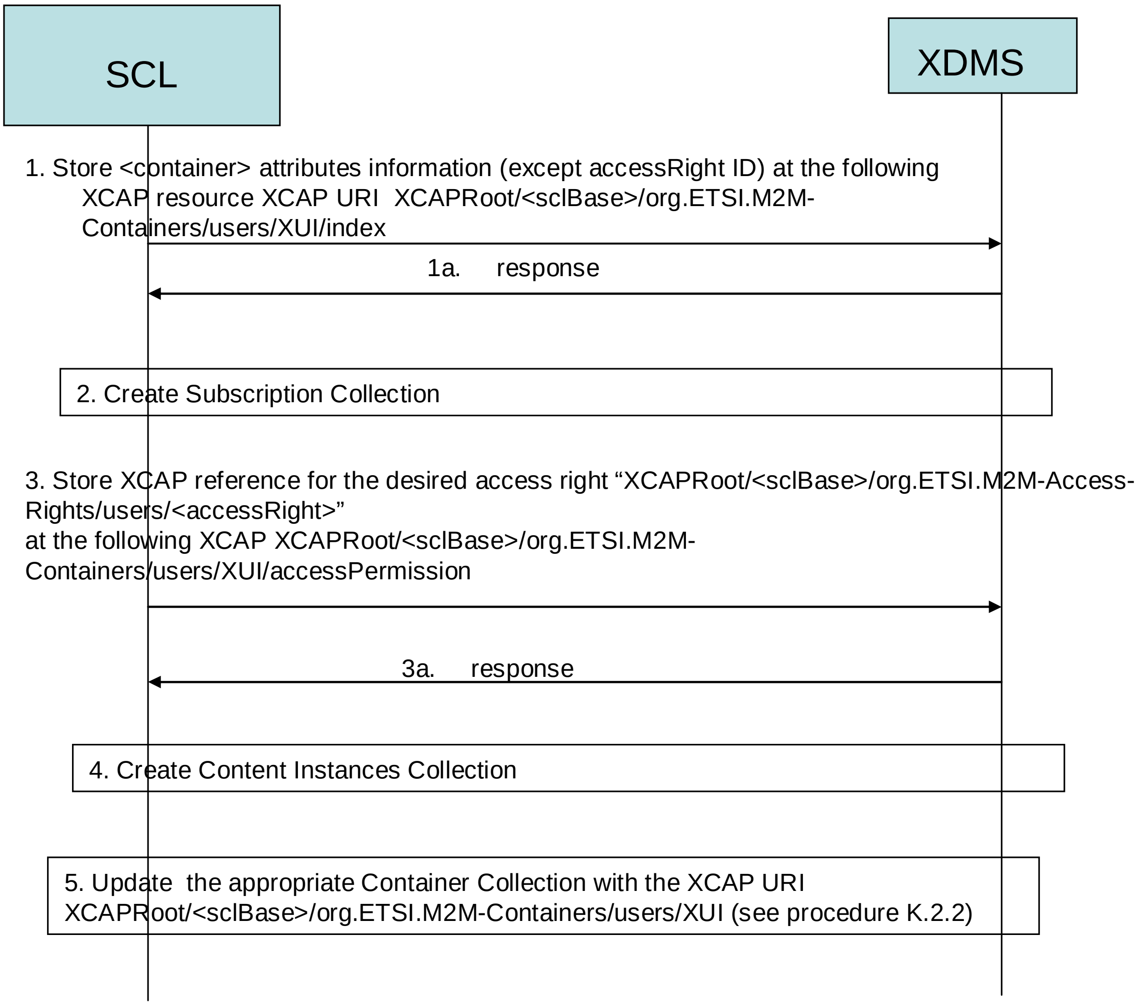

SCL
XDMS
1. Store <container> attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/XUI/accessPermission
3a. response
4. Create Content Instances Collection
5. Update the appropriate Container Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/XUI (see procedure K.2.2)

Figure F.39: Container Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the container in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Containers/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4 NSCL creates a Content Instance collection resource using procedure (clause F.2.1.20) for that matter.
- The NSCL updates the appropriate Container Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.7 Creation of m2mPoC Resource

Figure F.40 shows the procedure to be performed by the NSCL to create an m2mPoc XDM resource.

The m2mPoc Application Usage applies in the case a newly created m2mPoc resource is added as a member of an m2mPoc collection resource.

The NSCL shall create a new tree representing the m2mPoc resource under the users' tree for the m2mPoc Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/) for the m2mPoc applying procedure (clause F.1.3.1) in that regard.

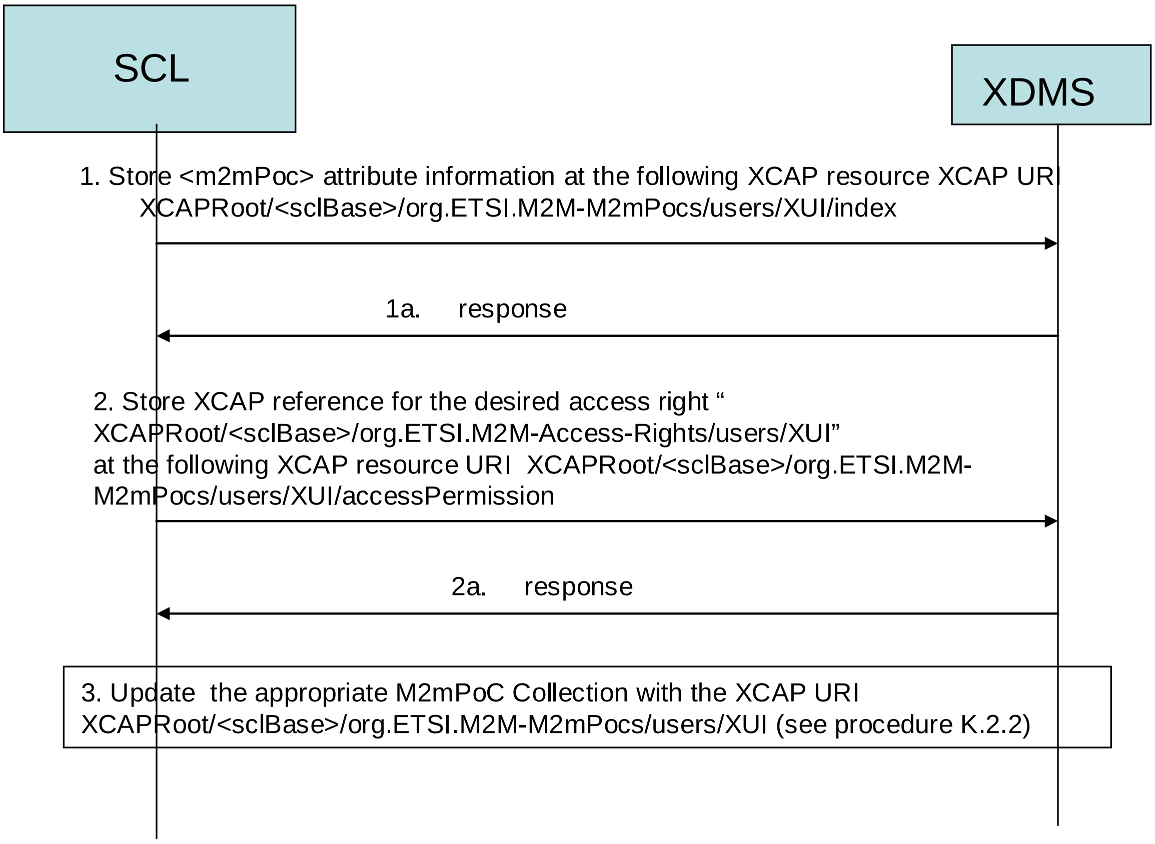

SCL
XDMS
1. Store <m2mPoc> attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/accessPermission
2a. response
3. Update the appropriate M2mPoC Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI (see procedure K.2.2)

Figure F.40: m2mPoc Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the m2mpPoc resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate M2mPoC Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.8 Creation of NotificationChannel Resource

Figure F.41 shows the procedure to be performed by the NSCL to create a Notification Channel XDM resource.

The Notification Channel Application Usage applies in the case a newly created Notification Channel resource is added as a member of a Notification Channel collection resource.

The NSCL shall create a new tree representing the Notification Channel resource under the users' tree for the Notification Channel Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users/XUI/) for the Notification Channel applying procedure (clause F.1.3.1) in that regard.

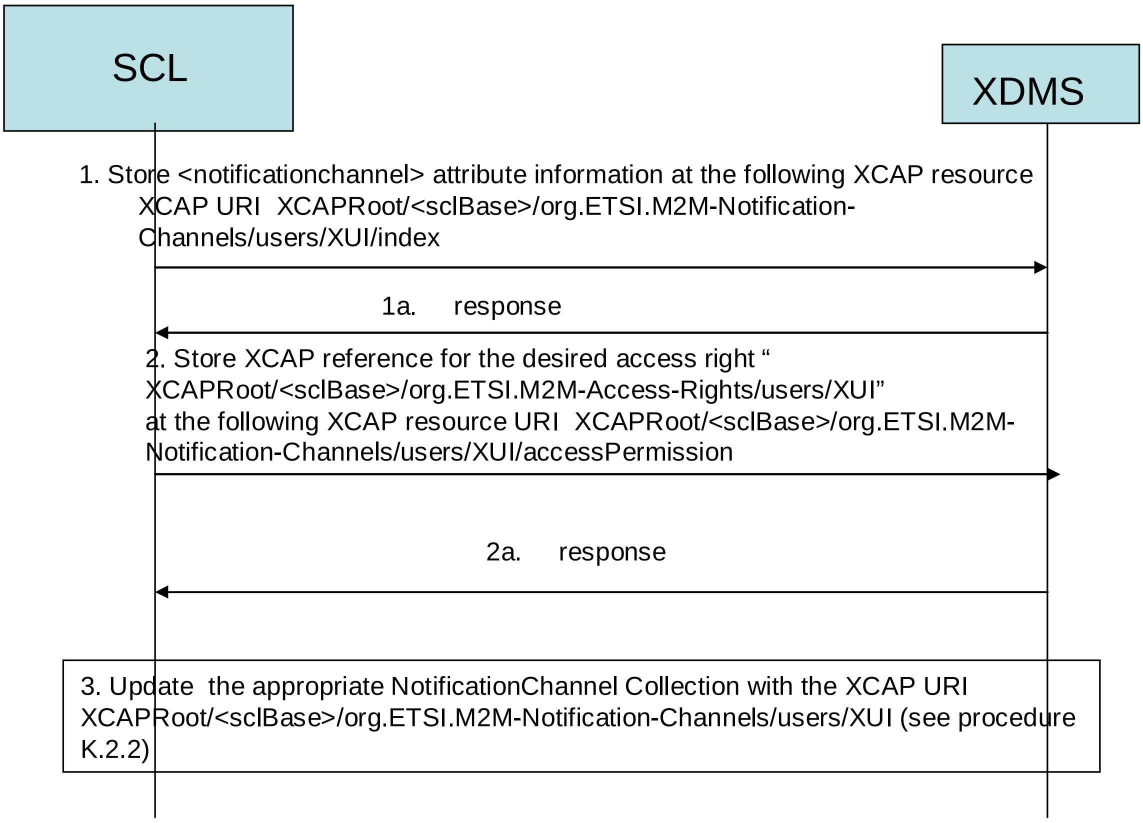

SCL
XDMS
1. Store <notificationchannel> attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users/XUI/accessPermission
2a. response
3. Update the appropriate NotificationChannel Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users/XUI (see procedure K.2.2)

Figure F.41: Notification Channel Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Notification Channel resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Notification-Channels/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate NotificationChannel Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.9 Creation of Location Container Resource

Figure F.42 shows the procedure to be performed by the NSCL to create a Location Container XDM resource.

The Location Container Application Usage applies in the case a newly created location container resource is added as a member of a container collection resource that belongs to any of the following M2M resources:

- Case 1: The container collection resource belongs to a registered SCL.
- Case 2: The container collection resource belongs to the sclBase.
- Case 3: The container collection resource belongs to a registered local application.
- Case 4: The container collection resource belongs to an announced application.

For all the above cases, the NSCL shall create a new tree representing the location container resource under the users' tree for the Location Container Application Usage. The M2M NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Location-Container/users/XUI/) for the container applying procedure (clause F.1.3.1) in that regard.

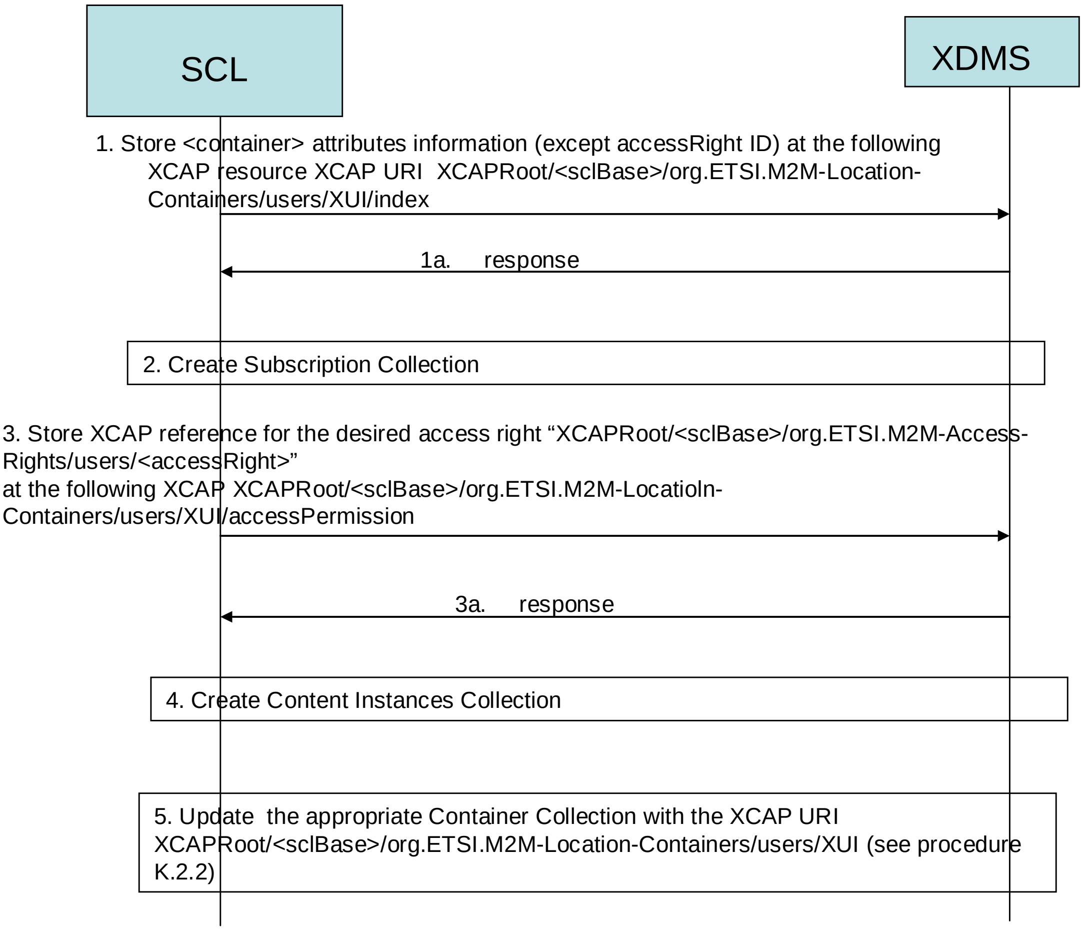

SCL
XDMS
1. Store <container> attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Location-Containers/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-LocatioIn-Containers/users/XUI/accessPermission
3a. response
4. Create Content Instances Collection
5. Update the appropriate Container Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Location-Containers/users/XUI (see procedure K.2.2)

Figure F.42: Location Container Resource creation process

The call flow is identical to that for a container resource and as such will not be repeated again.

F.2.1.10 Creation of Content Instance Resource

Figure F.43 shows the procedure to be performed by the NSCL to create a Content Instance XDM resource.

The Content Instance Application Usage applies in the case a newly created Content Instant resource is added as a member of a Content Instance collection resource.

The NSCL shall create a new tree representing the Content Instance resource under the users' tree for the Content Instance Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI/) for the Content Instance applying procedure (clause F.1.3.1) in that regard.

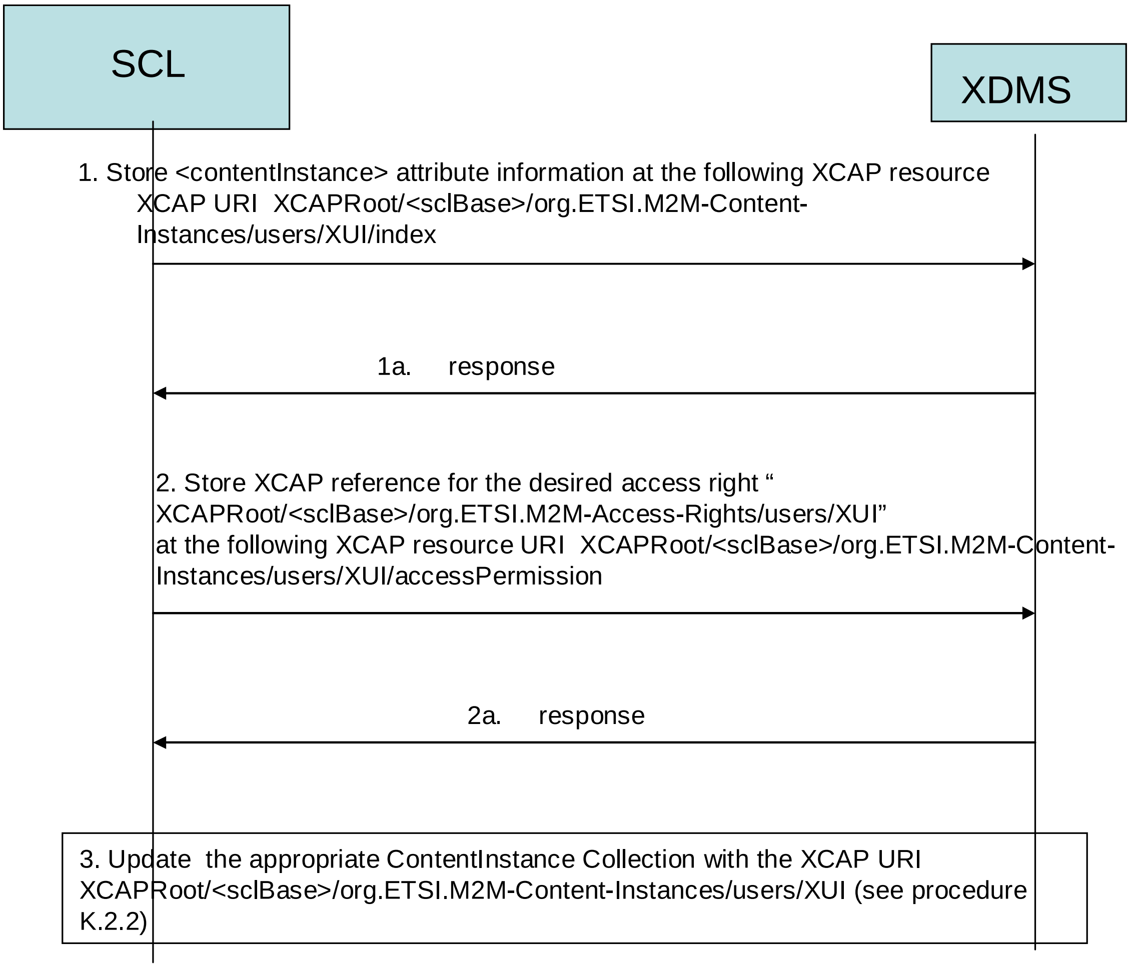

SCL
XDMS
1. Store <contentInstance> attribute information at the following XCAP resource
XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI/accessPermission
2a. response
3. Update the appropriate ContentInstance Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI (see procedure K.2.2)

Figure F.43: Content Instance Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Content Instance resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Content-Instances/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate ContentInstance Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.11 Creation of Subscription Resource

Figure F.44 shows the procedure to be performed by the NSCL to create a Subscription XDM resource.

The Subscription Application Usage applies in the case a newly created Subscription resource is added as a member of a Subscription collection resource.

The NSCL shall create a new tree representing the Subscription resource under the users' tree for the Subscription Application Usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users/XUI/) for the Subscription applying procedure (clause F.1.3.1) in that regard.

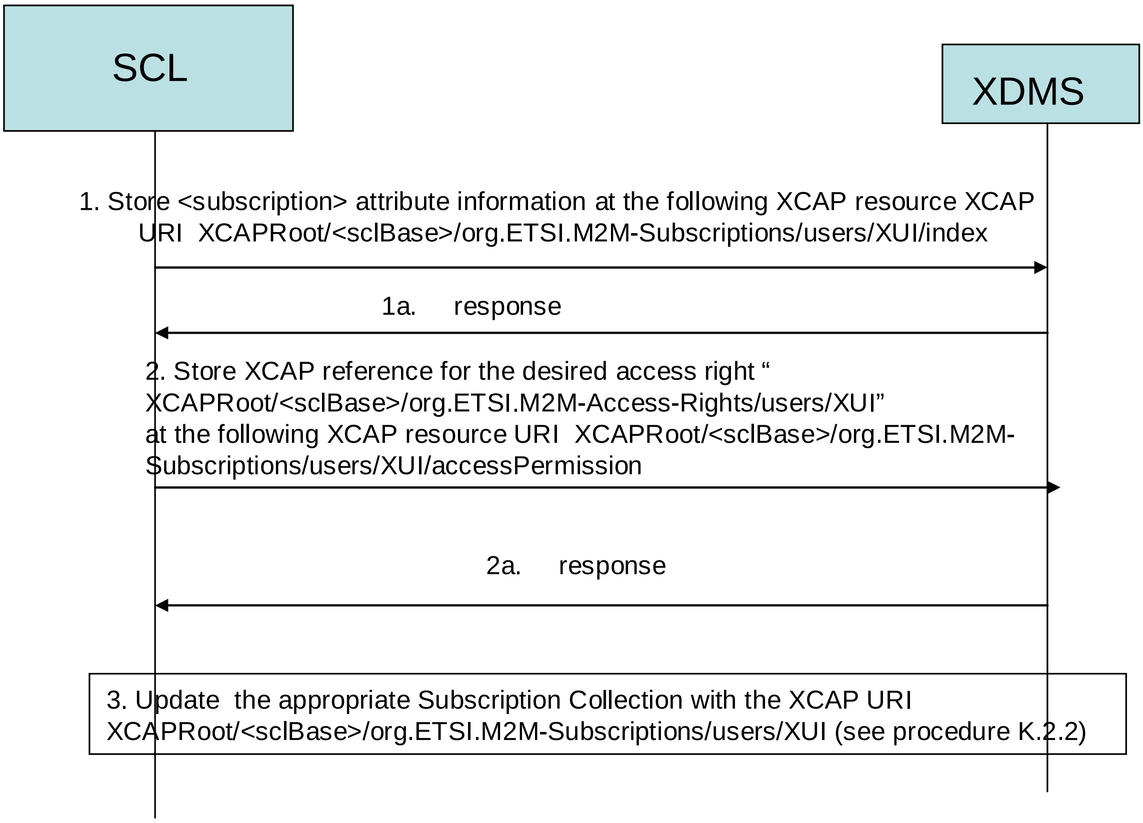

SCL
XDMS
1. Store <subscription> attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users/XUI/accessPermission
2a. response
3. Update the appropriate Subscription Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscriptions/users/XUI (see procedure K.2.2)

Figure F.44: Subscription Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPocs/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Subscription resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate Subscription Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.12 Creation of Group Collection

Figure F.45 shows the procedure to be performed by the NSCL to create a group collection XDM resource.

Group collections are created by the NSCL during the following instances:

- Case 1: At M2M NSCL start up.
- Case 2: A new SCL is registered.
- Case 3: A new application is registered.

- Case 4: A new application is announced.

For all the above cases, the NSCL shall create a new tree representing the group collection under the users' tree. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/) for the group collection applying procedure (clause F.1.3.1.1) for that purpose.

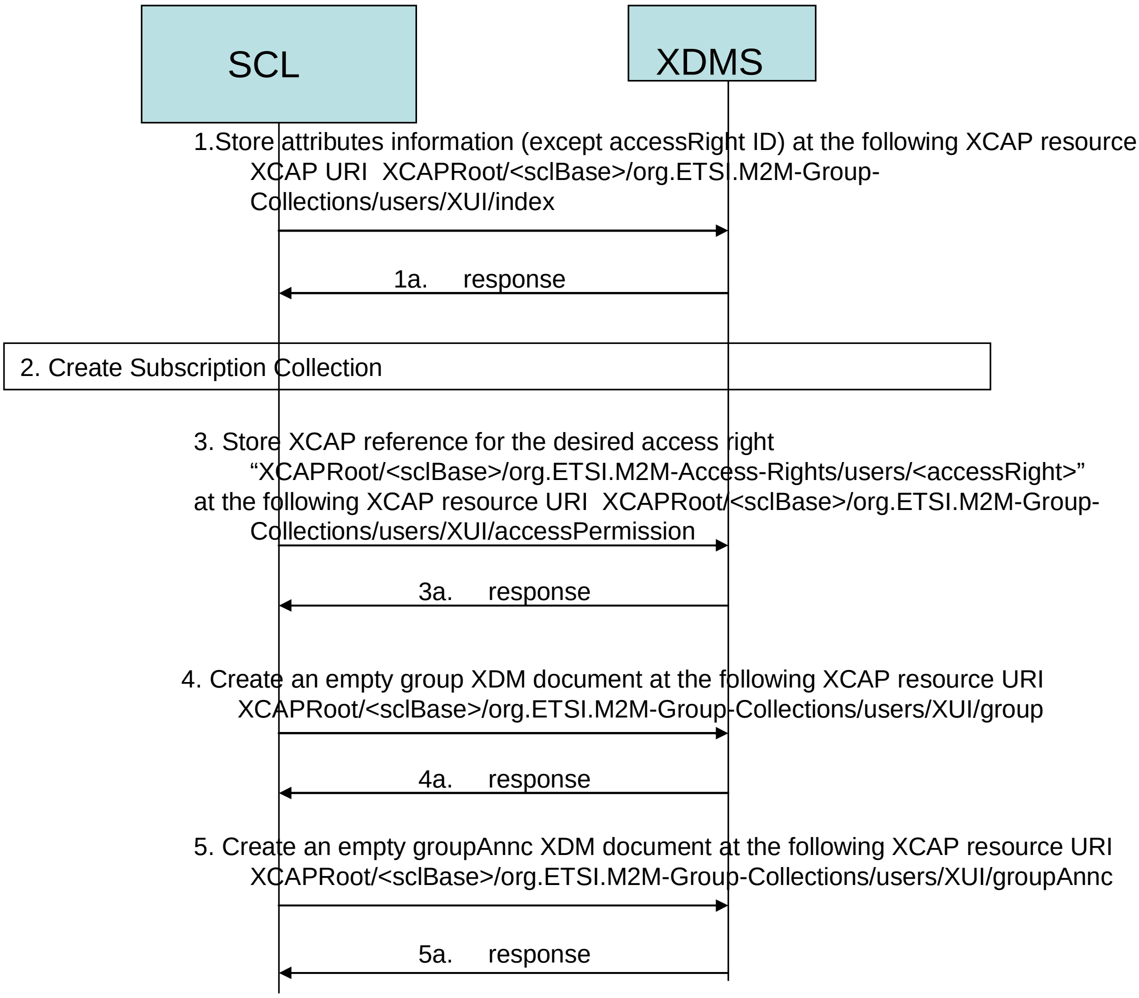

SCL
XDMS
1.Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/accessPermission
3a. response
4. Create an empty group XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/group
4a. response
5. Create an empty groupAnnc XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/groupAnnc
5a. response

Figure F.45: Group Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the group collection in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the M2M NSCL shall have the XCAP resource for it.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4, the M2M NSCL creates an empty document that holds all groups belonging to the group collection in the group XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/group.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, the M2M NSCL creates an empty document that holds all groupAnnc instances belonging to the group collection in the groupAnnc XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Collections/users/XUI/group.
- In step 5a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.13 Creation of Container Collection

Figures F.46 and F.47 shows the procedure to be performed by the NSCL to create a container collection XDM resource.

Container collections are created by the NSCL during the following instances:

- Case 1: At M2M NSCL start up.
- Case 2: A new SCL is registered.
- Case 3: A new application is registered.
- Case 4: A new application is announced.

For all the above cases, the NSCL shall create a new tree representing the container collection under the users' tree. The M2M NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/) for the container collection applying procedure (clause F.1.3.1.1).

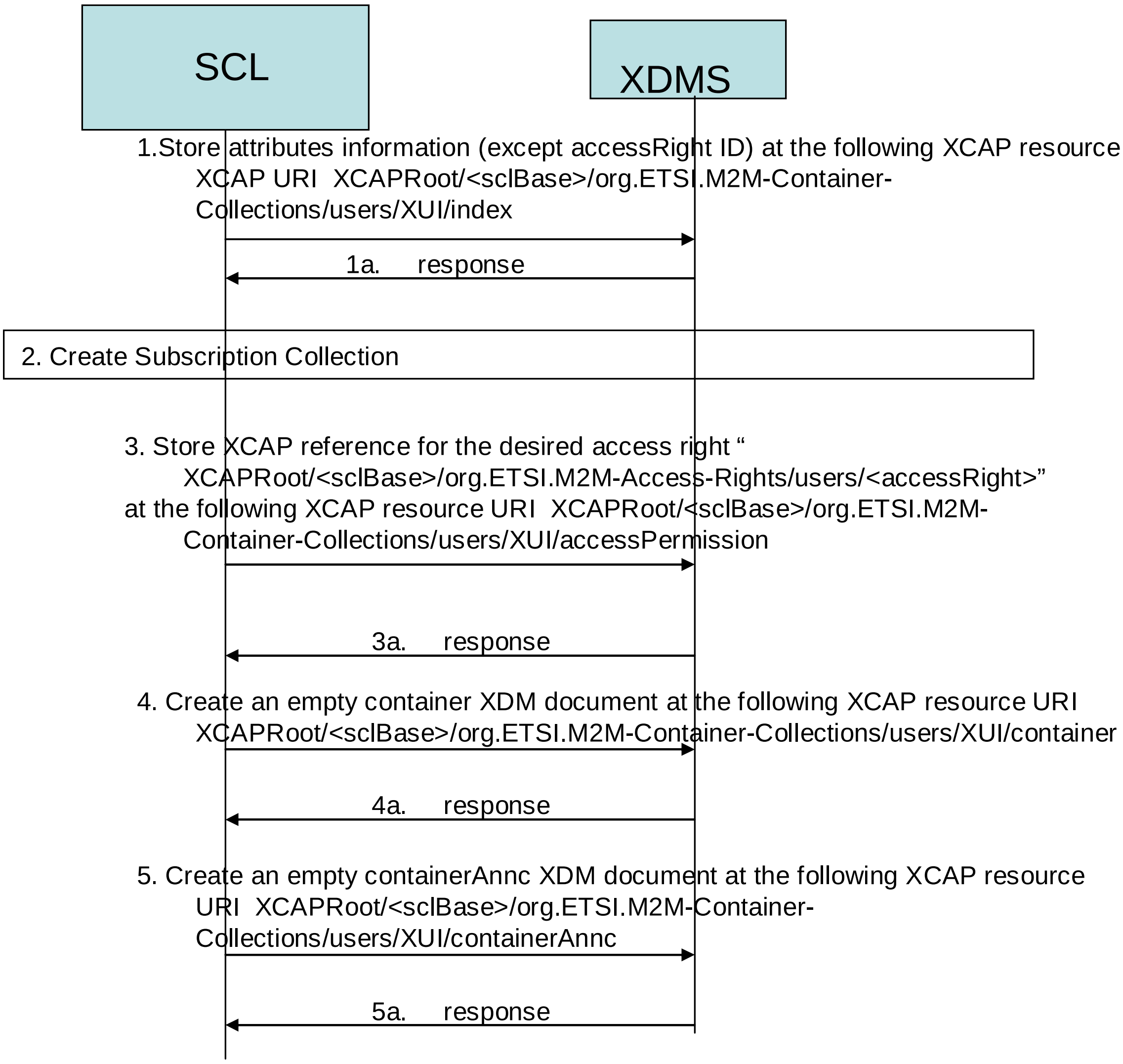

SCL
XDMS
1.Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/accessPermission
3a. response
4. Create an empty container XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/container
4a. response
5. Create an empty containerAnnc XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/containerAnnc
5a. response

Figure F.46: Container Collection Resource creation process - part 1

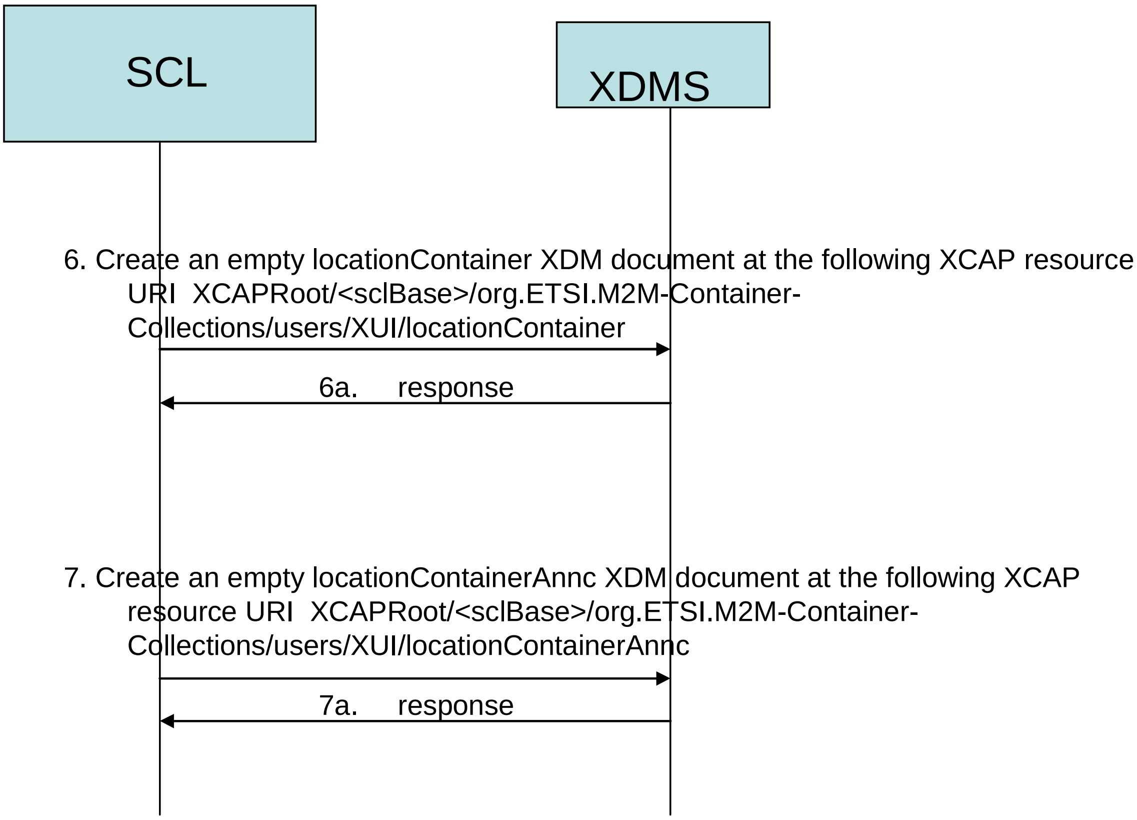

SCL
XDMS
6. Create an empty locationContainer XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/locationContainer
6a. response
7. Create an empty locationContainerAnnc XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/locationContainerAnnc
7a. response

Figure F.47: Container Collection Resource creation process - part 2

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the container collection in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the NSCL creates an empty document that holds all containers belonging to the container collection in the container XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/container.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, the NSCL creates an empty document that holds all containerAnnc resources belonging to the container collection in the containerAnnc XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/containerAnnc.
- In step 5a, XDMS acknowledges the successful storage of the XDM document.
- In step 6, the NSCL creates an empty document that holds all locationContainer resources belonging to the container collection in the container XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/locationContainer.
- In step 6a, XDMS acknowledges the successful storage of the XDM document.

- In step 7, the NSCL creates an empty document that holds all locationContainerAnnc resources belonging to the container collection in the containerAnnc XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Container-Collections/users/XUI/locationContainerAnnc.
- In step 7a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.15 Creation of Access Right Collection

Figure F.48 shows the procedure to be performed by the NSCL to create an access right collection XDM resource.

Access right collections are created by the NSCL during the following instances:

- Case 1: At M2M NSCL start up.
- Case 2: A new SCL is registered.
- Case 3: A new application is registered.
- Case 4: A new application is announced.

For all the above cases, the NSCL shall create a new tree representing the access right collection under the users' tree. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/) for the access right collection applying procedure (clause F.1.3.1.1).

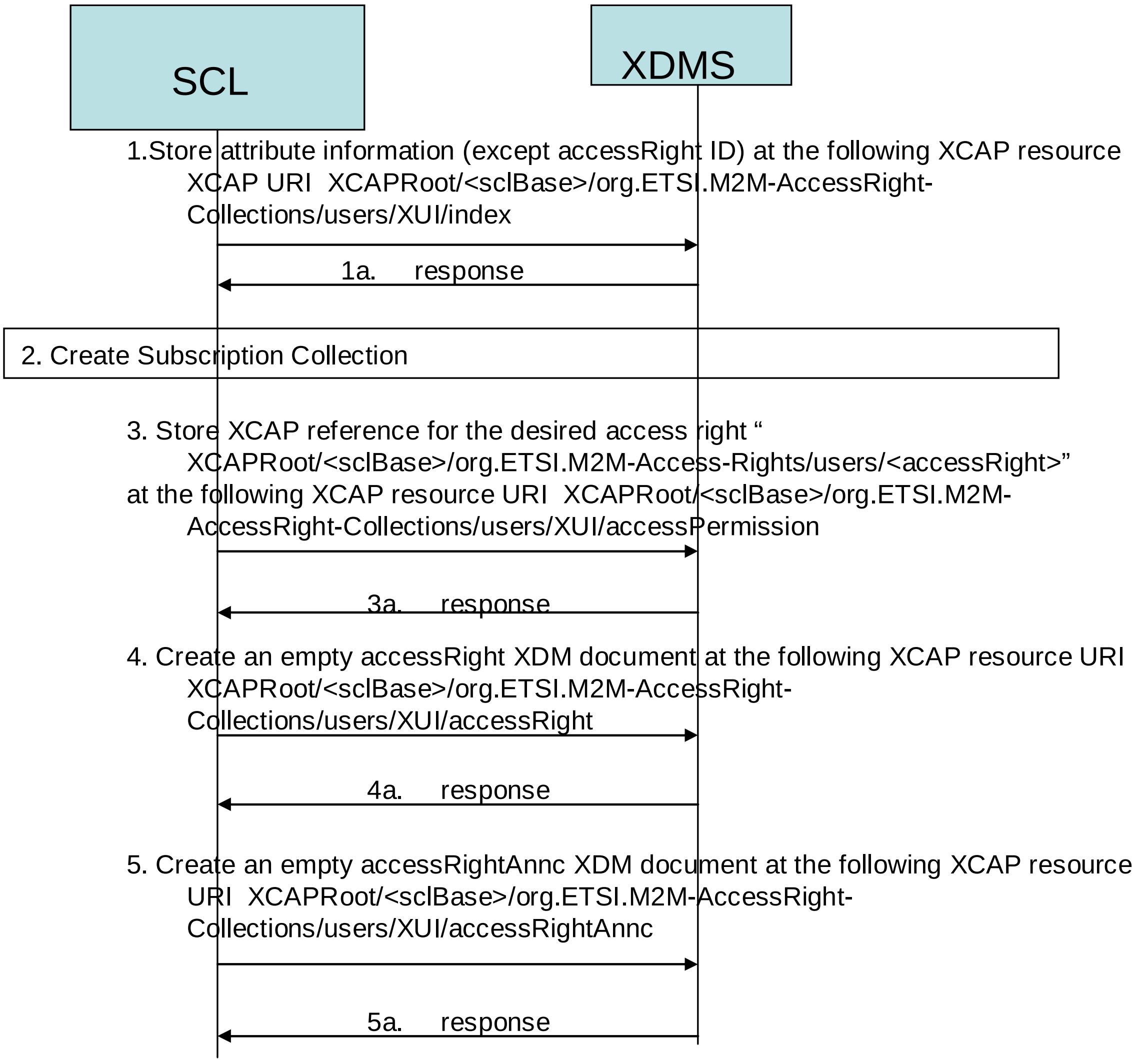

SCL
XDMS
1.Store attribute information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/accessPermission
3a. response
4. Create an empty accessRight XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/accessRight
4a. response
5. Create an empty accessRightAnnc XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/accessRightAnnc
5a. response

Figure F.48: Access Right Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Access Right-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the access right collection in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the NSCL creates an empty document that holds all access right resources belonging to the access right collection in the access right XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/XUI/accessRight.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, the NSCL creates an empty document that holds all accessRightAnnc resources belonging to the access right collection in the accessRightAnnc XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Collections/users/<ARCollection>/accessRightAnnc.
- In step 5a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.16 Creation of Application Collection

Figure F.49 shows the procedure to be performed by the M2M NSCL to create an application collection XDM resource.

Application collections are created by an M2M NSCL during the following instances:

- At M2M NSCL start up.
- A new SCL is registered.

For both of the above cases, the M2M NSCL shall create a new tree representing the application collection under the users' tree. The M2M NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/) for the application collection applying procedure (clause F.1.3.1.1).

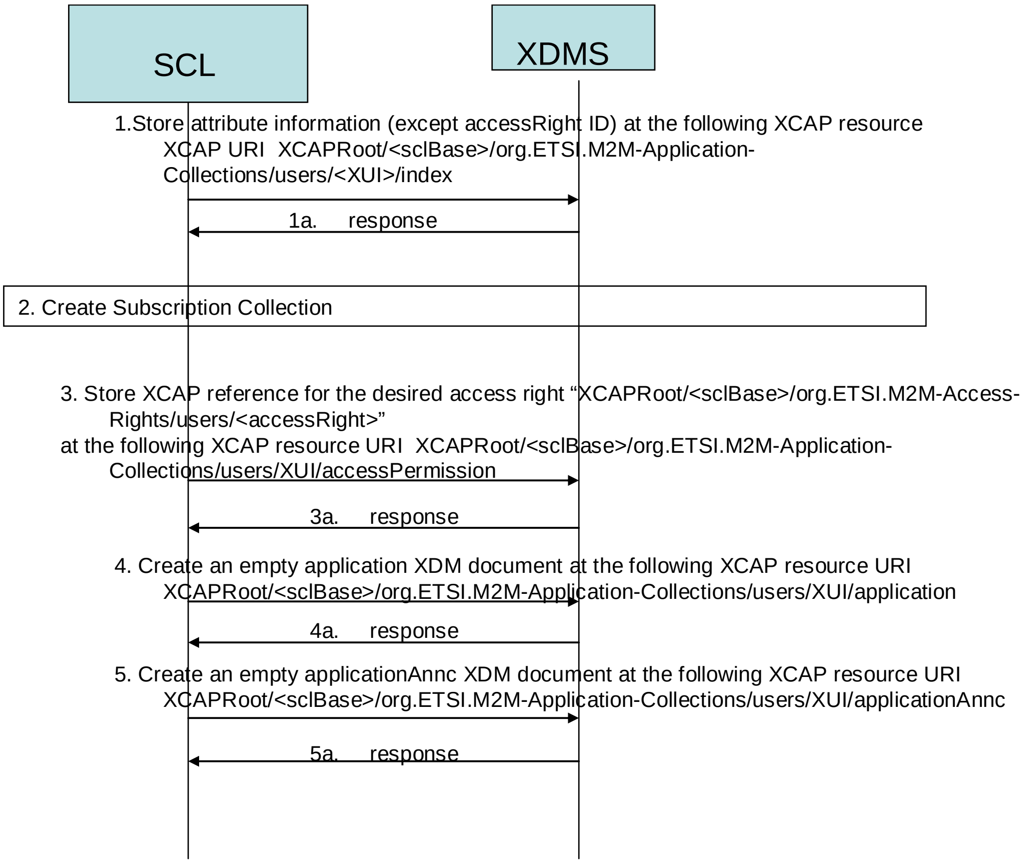

SCL
XDMS
1.Store attribute information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/<XUI>/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/accessPermission
3a. response
4. Create an empty application XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/application
4a. response
5. Create an empty applicationAnnc XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/applicationAnnc
5a. response

Figure F.49: Application Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the application collection in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4, the NSCL creates an empty document that holds all applications belonging to the application collection in the application XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/application.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.

- In step 5, the NSCL creates an empty document that holds all applicationAnnc resources belonging to the application collection in the applicationAnnc XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Application-Collections/users/XUI/application.

- In step 5a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.17 Creation of SCL Collection

Figure F.50 shows the procedure to be performed by the NSCL to create an scl collection XDM resource.

SCL collections are created by the NSCL at NSCL start up. Note that there is only one SCL collection created by the NSCL.

In this case, the NSCL creates a new tree representing the SCL collection resource under the users' tree as per the SCL Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/XUI/) for the SCL collection applying procedure (clause F.1.3.1.1).

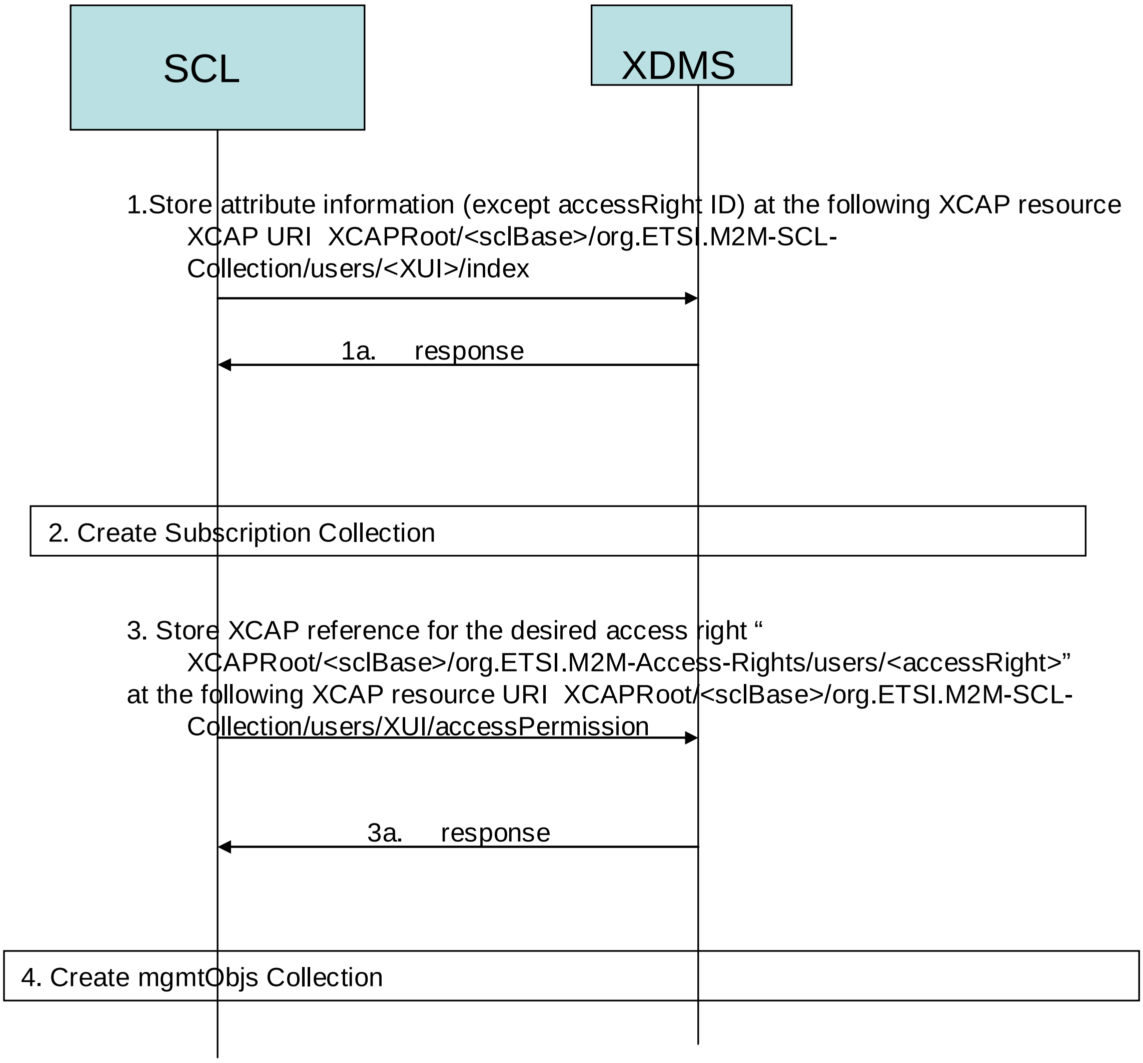

SCL
XDMS
1.Store attribute information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/<XUI>/index
1a. response
2. Create Subscription Collection
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/XUI/accessPermission
3a. response
4. Create mgmtObjs Collection

Figure F.50: SCL Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the <sclCollection> in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-CL-Collection/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.18 Creation of m2mPoc Collection

Figure F.51 shows the procedure to be performed by the NSCL to create an m2mPoc collection XDM resource.

In this case, the NSCL creates a new tree representing the m2mPoc collection resource under the users' tree as per the m2mPoc Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collection/users/XUI/) for the m2mPoc collection applying procedure (clause F.1.3.1.1). This results in the XUI being the same as the parent resource for the collection.

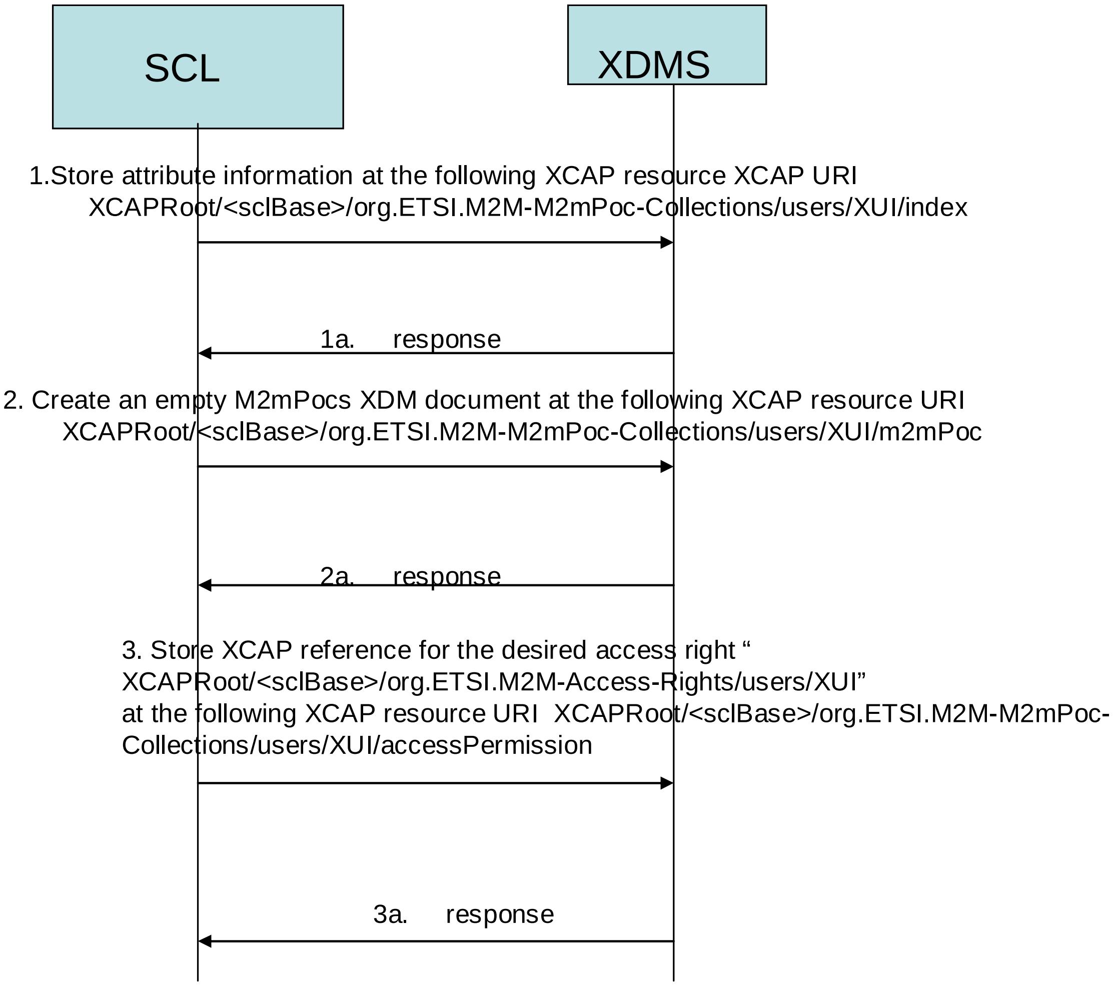

SCL
XDMS
1.Store attribute information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/index
1a. response
2. Create an empty M2mPocs XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/m2mPoc
2a. response
3. Store XCAP reference for the desired access right "
XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/accessPermission
3a. response

Figure F.51: m2mPoc Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates an empty XML document and store it at the following XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/m2mPoc.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- In step 3, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the m2mPoc Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.19 Creation of Notification Channel Collection

Figure F.52 shows the procedure to be performed by the NSCL to create a Notification Channel collection XDM resource.

In this case, the NSCL creates a new tree representing the Notification Channel collection resource under the users' tree as per the Notification Channel Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collection/users/XUI/) for the Notification Channel collection applying procedure (clause F.1.3.1.1).

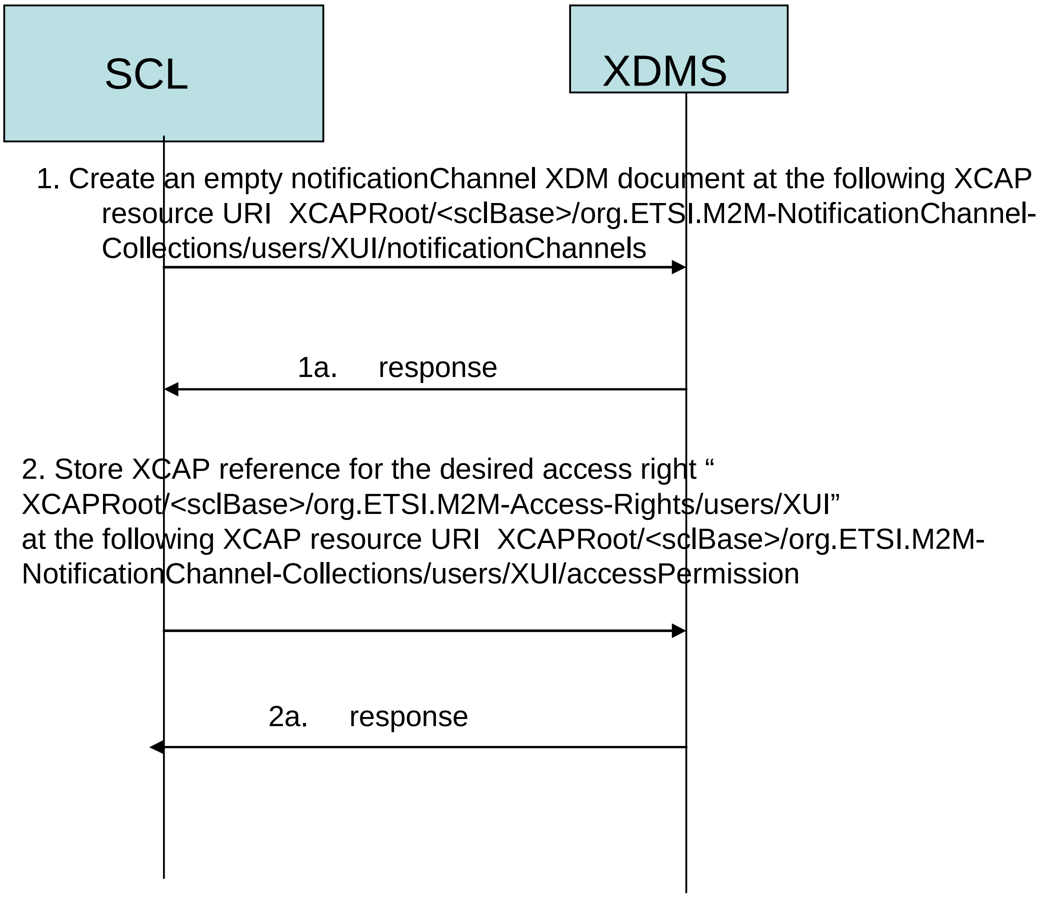

SCL
XDMS
1. Create an empty notificationChannel XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collections/users/XUI/notificationChannels
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collections/users/XUI/accessPermission
2a. response

Figure F.52: Notification Channel Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, NSCL creates an empty XML document and store it at the following XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collections/users/XUI/notificationChannels.

- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Notification Channel Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-NotificationChannel-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 2a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.20 Creation of Subscription Collection

Figure F.53 shows the procedure to be performed by the NSCL to create a Subscription collection XDM resource.

In this case, the NSCL creates a new tree representing the Subscription collection resource under the users' tree as per the Subscription Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users/XUI/) for the Subscription collection applying procedure (clause F.1.3.1.1).

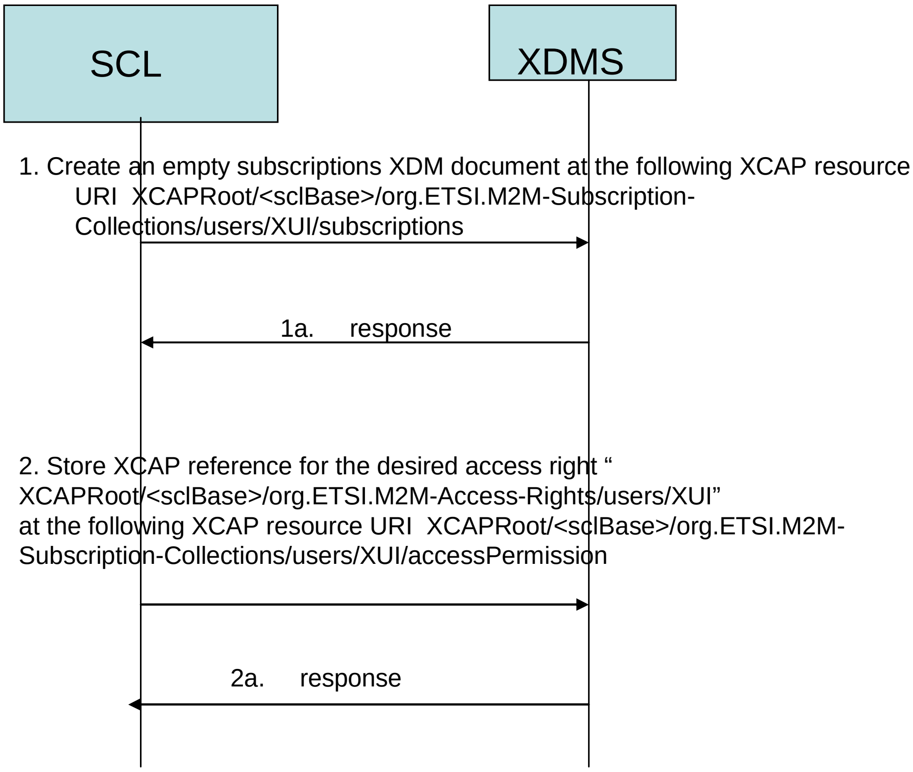

SCL
XDMS
1. Create an empty subscriptions XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users/XUI/subscriptions
1a. response
2. Store XCAP reference for the desired access right “ XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI” at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users/XUI/accessPermission
2a. response

Figure F.53: Subscription Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, NSCL creates an empty XML document and store it at the following XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users/XUI/subscriptions.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Subscription Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Subscription-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.21 Creation of ContentInstance Collection

Figure F.54 shows the procedure to be performed by the NSCL to create a Content Instance collection XDM resource.

In this case, the NSCL creates a new tree representing the Content Instance collection resource under the users' tree as per the Content Instance Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M- ContentInstance-Collection/users/XUI/) for the Content Instance collection applying procedure (clause F.1.3.1.1).

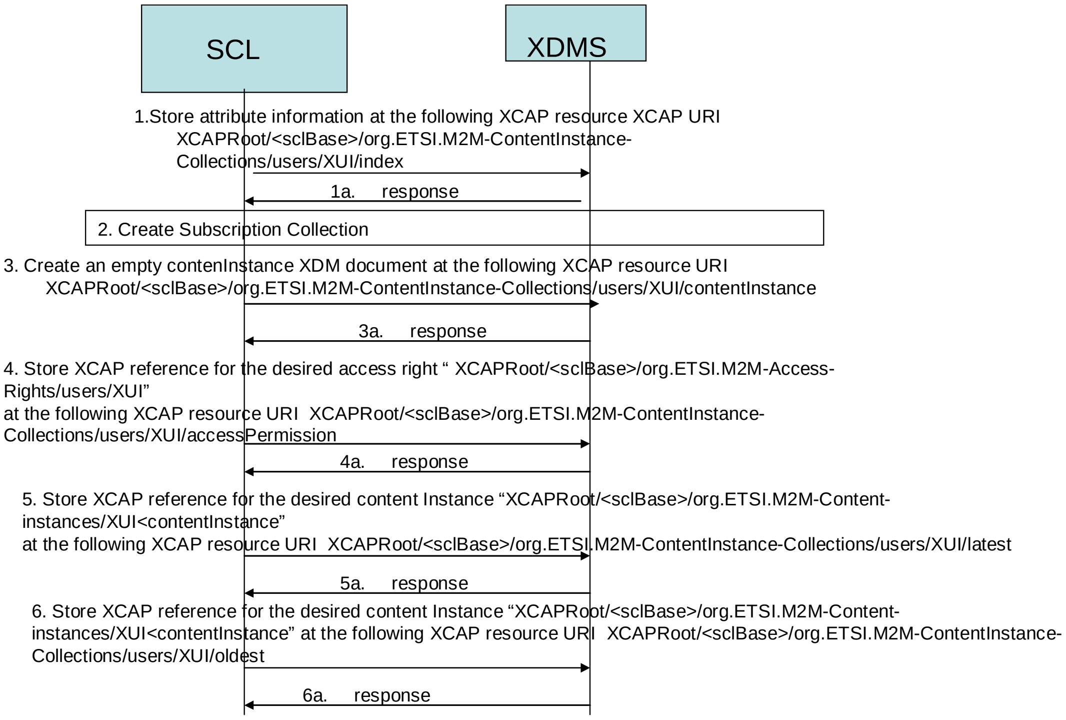

SCL
XDMS
1.Store attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty contenInstance XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/contentInstance
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/accessPermission
4a. response
5. Store XCAP reference for the desired content Instance "XCAPRoot/<sclBase>/org.ETSI.M2M-Content-instances/XUI<contentInstance" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/latest
5a. response
6. Store XCAP reference for the desired content Instance "XCAPRoot/<sclBase>/org.ETSI.M2M-Content-instances/XUI<contentInstance" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/oldest
6a. response

Figure F.54: Content Instance Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, NSCL creates an empty XML document and store it at the following XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-M2mPoc-Collections/users/XUI/m2mPoc.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Content Instance Channel Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, the NSCL stores the XCAP resource that identifies the content Instance resource that applies to the Content Instance Collection attribute "latest" in the latest XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/latest.
- In step 5a, XDMS acknowledges the successful storage of the XDM document.
- In step 6, the NSCL stores the XCAP resource that identifies the content Instance resource that applies to the Content Instance Collection attribute "oldest" in the oldest XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ContentInstance-Collections/users/XUI/oldest.
- In step 6a, XDMS acknowledges the successful storage of the XDM document.

F.2.1.22 Creation of Announced Group Resource

Figures F.55 show the procedure to be performed by the NSCL to create an Announced Group XDM resource.

The Announced Group Application Usage applies in the case an announced group resource is added as member of a group collection resource that belongs to a registered SCL.

In this case, the NSCL shall create a new tree representing the announced group resource under the users' tree for the Announced Group "Application Usage". The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/XUI/) for the announced group applying procedure (clause F.1.3.1) in that regard.

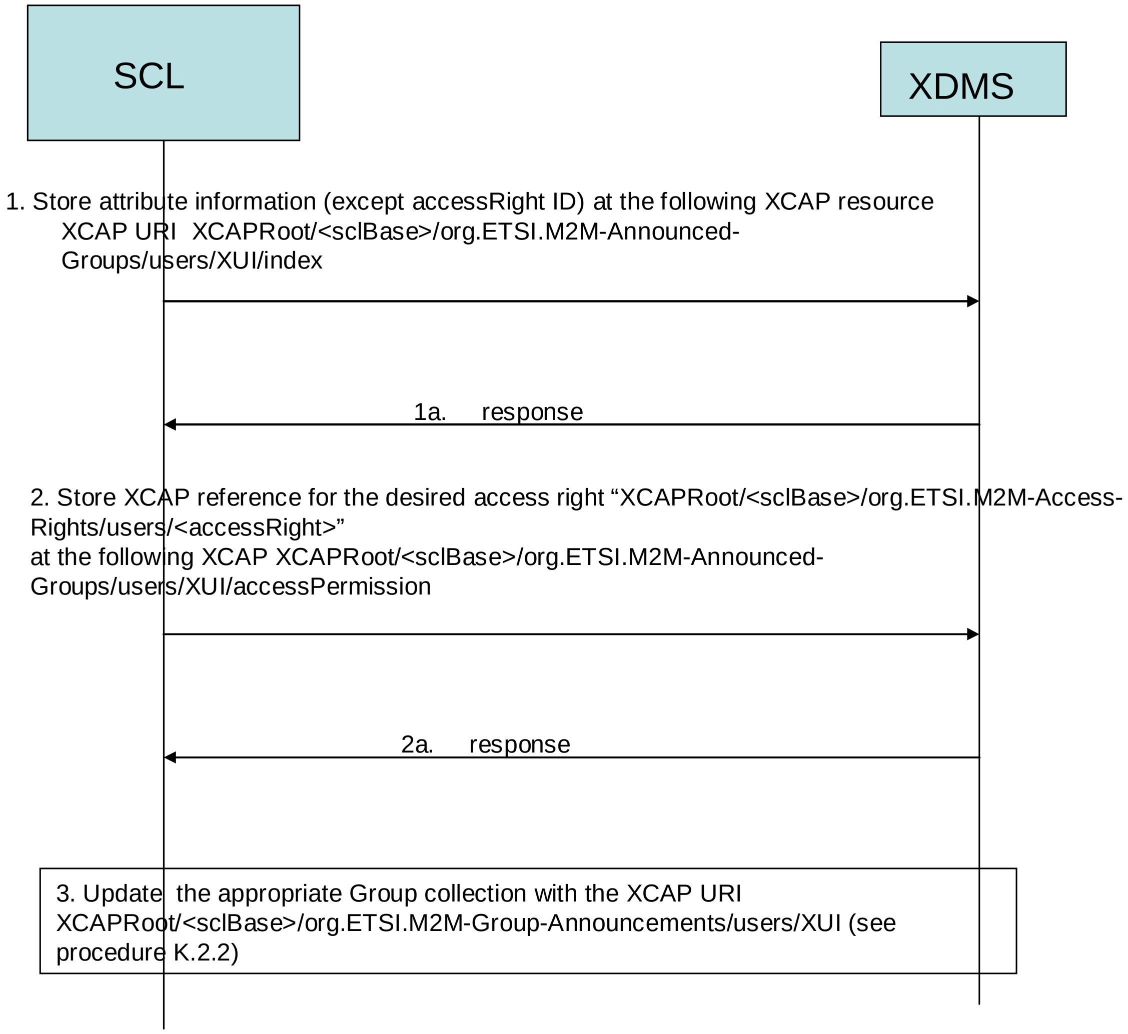

SCL
XDMS
1. Store attribute information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right “XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>” at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/XUI/accessPermission
2a. response
3. Update the appropriate Group collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Group-Announcements/users/XUI (see procedure K.2.2)

Figure F.55: Announced Group Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the groupAnnc in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Groups/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate Group Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.23 Creation of Announced Container Resource

Figure F.56 show the procedure to be performed by the NSCL to create an Announced Container XDM resource.

The Announced Container Application Usage applies in the case an announced container resource is added as member of a container collection resource that belongs to a registered SCL.

In this case, the NSCL shall create a new tree representing the announced container resource under the users' tree for the Announced Container "Application Usage". The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Containers/users/XUI/) for the announced container applying procedure (clause F.1.3.1) in that regard.

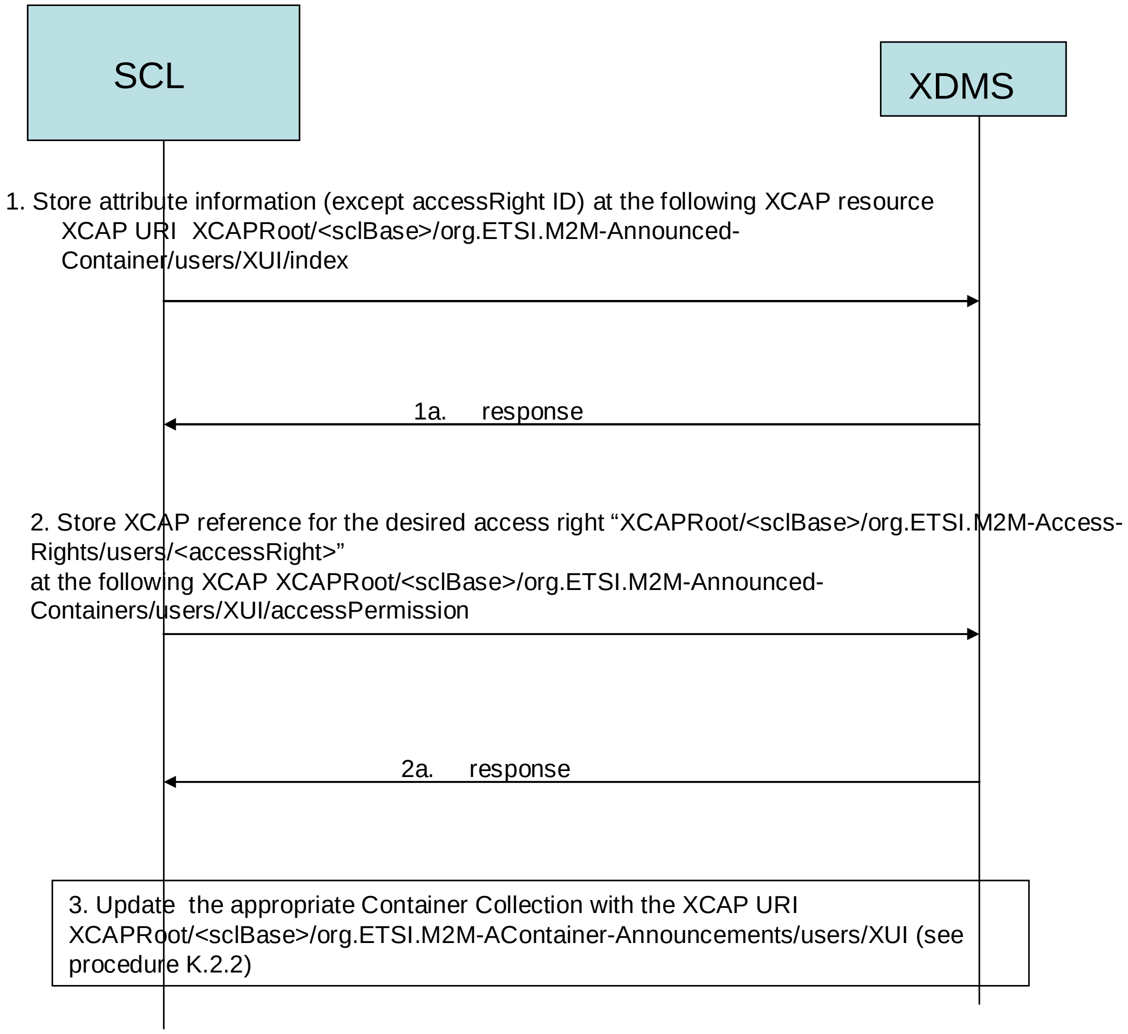

SCL
XDMS
1. Store attribute information (except accessRight ID) at the following XCAP resource
XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Container/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right “XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>”
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Containers/users/XUI/accessPermission
2a. response
3. Update the appropriate Container Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AContainer-Announcements/users/XUI (see procedure K.2.2)

Figure F.56: Announced Container Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-Containers/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the containerAnnc in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced- Containers/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.

- In step 2a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate Container Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.24 Creation of Announced AccessRight Resource

Figures F.57 show the procedure to be performed by the NSCL to create an Announced accessRight XDM resource.

The Announced accessRight Application Usage applies in the case an announced accessRight resource is added as member of an access right collection resource that belongs to a registered SCL.

In this case, the NSCL shall create a new tree representing the announced accessRight resource under the users' tree for the Announced accessRight "Application Usage". The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Announced- AccessRights /users/XUI/) for the announced accessRight applying procedure (clause F.1.3.1) in that regard.

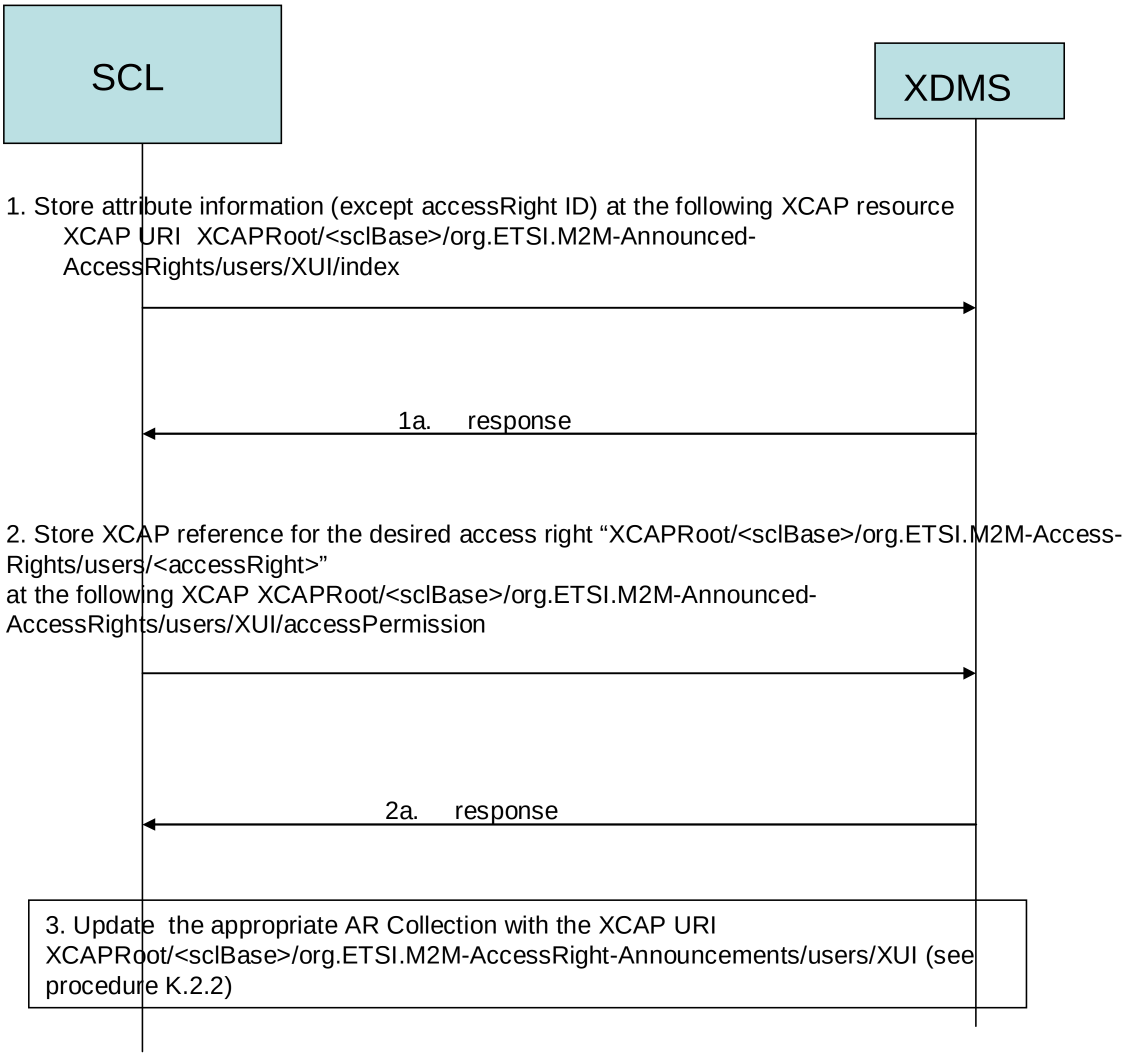

SCL
XDMS
1. Store attribute information (except accessRight ID) at the following XCAP resource
XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-AccessRights/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right “XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>”
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-AccessRights/users/XUI/accessPermission
2a. response
3. Update the appropriate AR Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AccessRight-Announcements/users/XUI (see procedure K.2.2)

Figure F.57: Announced Access Right Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-AccessRights/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the accessRightAnnc in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced- AccessRights/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate ARCollection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.25 Creation of Announced LocationContainer Resource

Figures F.58 show the procedure to be performed by the NSCL to create an Announced locationContainer XDM resource.

The Announced locationContainer Application Usage applies in the case an announced locationContainer resource is In this case, the NSCL shall create a new tree representing the announced locationContainer resource under the users' tree for the Announced locationContainer "Application Usage". The NSCL shall create the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Announced- LocationContainers /users/XUI/) for the announced locationContainer applying procedure (clause F.1.3.1) in that regard.

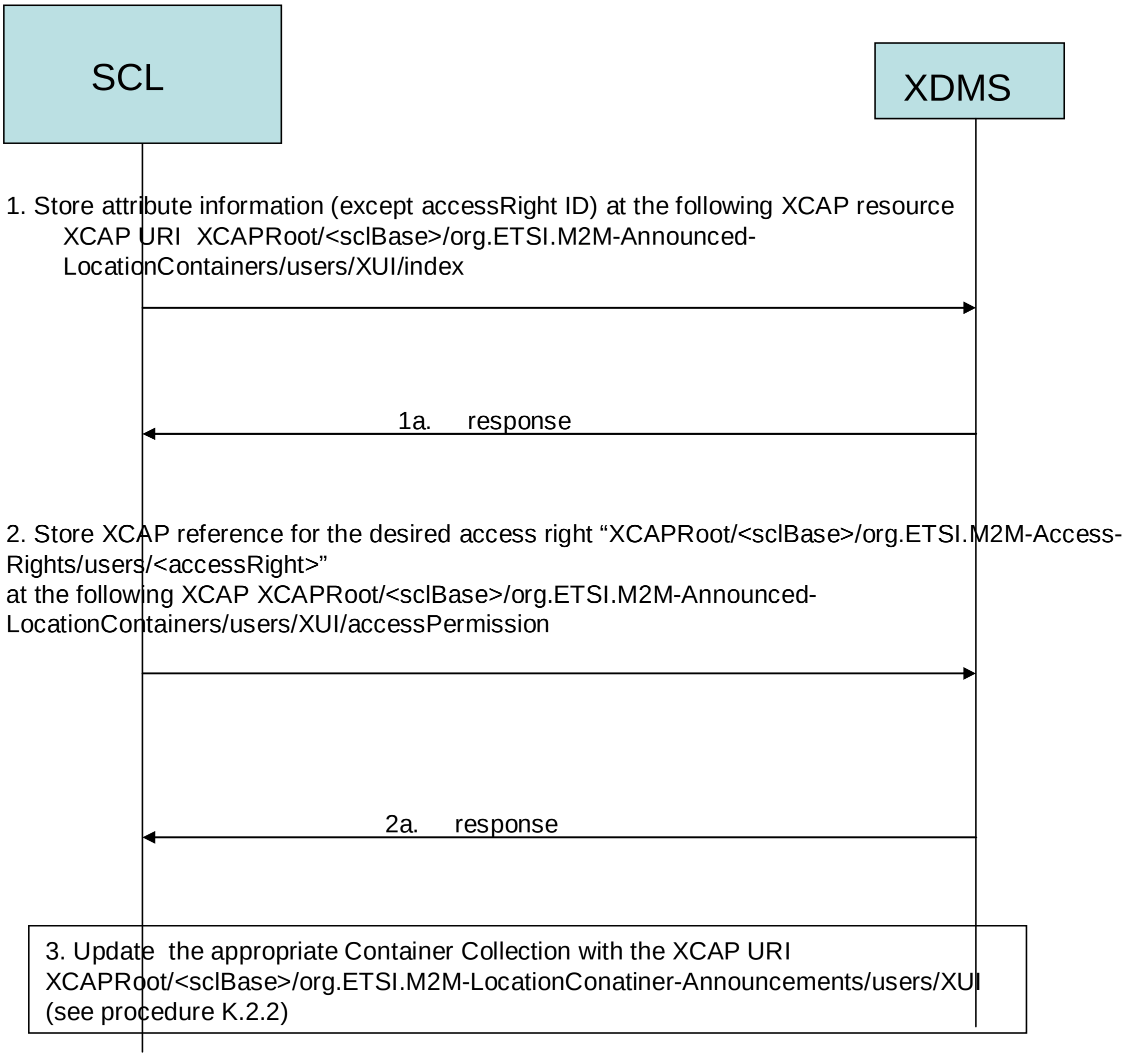

SCL
XDMS
1. Store attribute information (except accessRight ID) at the following XCAP resource
XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-LocationContainers/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-LocationContainers/users/XUI/accessPermission
2a. response
3. Update the appropriate Container Collection with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-LocationConatiner-Announcements/users/XUI
(see procedure K.2.2)

Figure F.58: Announced Location Container Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced-LocationContainers/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the locationContainerAnnc in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Announced- LocationContainers/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 2a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate Container Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.2.1.26 Resource Creation at System Startup

Figures F.59 show the procedure to be performed by the NSCL at system startup.

The SCLBase Application Usage applies in this case.

As a general principle at system startup all mandatory resources and sub-resources are created by the NSCL even if they are empty.

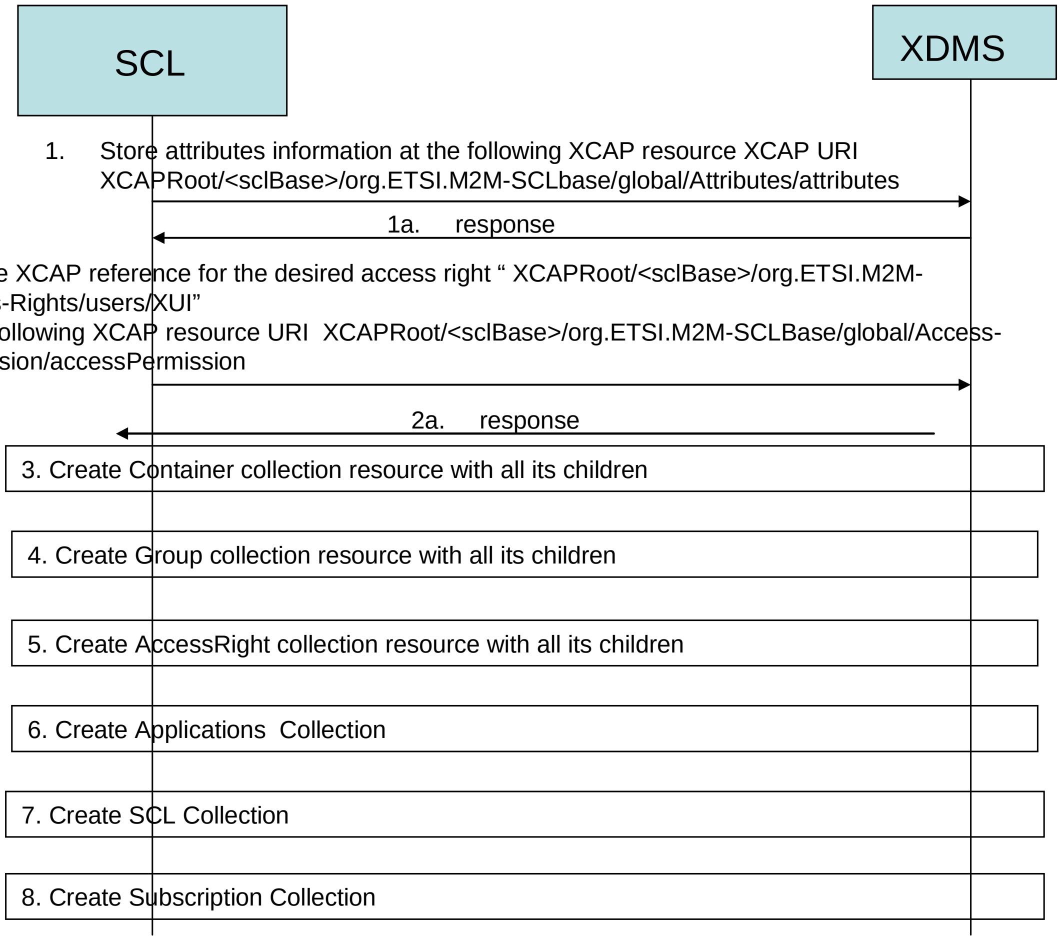

SCL
XDMS
1. Store attributes information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCLbase/global/Attributes/attributes
1a. response
2. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCLBase/global/Access-Permission/accessPermission
2a. response
3. Create Container collection resource with all its children
4. Create Group collection resource with all its children
5. Create AccessRight collection resource with all its children
6. Create Applications Collection
7. Create SCL Collection
8. Create Subscription Collection

Figure F.59: M2M SCP system Startup Procedure

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL creates stores all the sclBase attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCLBase/global/Attributes/attributes.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a container collection resource using procedure (clause F.2.1.13) for that matter.
- In step 3, NSCL creates a Group collection resource using procedure (clause F.2.1.12) for that matter.
- In step 4, NSCL creates an accessRight collection resource using procedure (clause F.2.1.14) for that matter.
- In step 5, NSCL creates an application collection resource using procedure (clause F.2.1.15) for that matter.
- In step 6, NSCL creates an SCL collection resource using procedure (clause F.2.1.16) for that matter.

- In step 7, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to all global resources in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCLBase/global/Access-Permission/accessPermission. Note that the accessRight resource should have been created before that and the NSCL shall have the XCAP resource for it.
- In step 7a, XDMS acknowledges the successful storage of the XDM document.
- In step 8, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

F.2.2 Update an existing Collection with a new Resource

This is a two step process, first the resource shall be created, and then the impacted collection list shall be updated with the new resource. Hence, the NSCL shall follow the following steps for updating all collections.

Step 1 - Create the XCAP URI for the new resource

Apply procedure (clause F.1.3.1.1) in that regard on the targetID. Note that the targetID in this case will include the new resource to be created.

Step 2 - Create the resource in XDMS using the applicable procedure defined above depending on the resource

E.g. container resource, group resource, accessRight resource, etc.

Step 3 - Read the collection document for updating it

The XCAP URI for the collection document shall be determined as described in clause F.1.3.1. To that effect, the targetID for the impacted collection shall be first identified. The original targetID shall be modified so that all elements starting from the collection name is removed. The NSCL then performs the procedure defined in clause F.1.3.1.1 on the modified targetID. The identified document is then read. Note that the XDM document name in this case will correspond to the group document to be read.

Step 4 - The NSCL shall update fetched document with the XCAP URI of the created resource in step 1

Step 5 - The NSCL shall then write back the XDM document to XDMS using the appropriate XDMS procedures

F.2.3 Writing/Updating data from an existing resource (not in a Collection)

The NSCL shall perform the following steps.

Step 1 - Create the XCAP URI for the Resource

Apply procedure (clause F.1.3.1.1) in that regard.

Step 2 - Read from XDMS the appropriate XDM document corresponding to the named resource

The NSCL shall read the XDM document corresponding to the last element in the incoming M2M URI using the appropriate XDMS procedure. The tables in identified application usage shall be used to select the XDM document.

Step 3 - The NSCL shall update the fetched document

Step 4 - The NSCL shall then write back the XDM document to XDMS using the appropriate XDMS procedures

F.2.4 Deleting an existing resource

The NSCL shall perform the following steps.

Step 1 - Create the XCAP URI for the Resource

Apply procedure (clause F.1.3.1.1) in that regard.

Step 2 - Delete the Resource in XDMS

The resource shall be deleted in XDMS using the proper application usage and XDMS procedures in that regard.

Step 3 - If the deleted resource belongs to a collection, then the collection has to be updated

The NSCL shall perform steps 3-4 of clause F.2.2 to update the collection.

F.3 NSCL Support for Delegation of M2M Subscriptions to the XDMS Subscription framework

F.3.1 Creating a subscription

Figure F.60 shows a call flow for the procedure to be performed by the NSCL for delegating management of subscriptions to XDMS.

The following is a brief description of the steps in the call flow:

- In step 1 the NSCL receives a request to create a subscription to an M2M resource.
- In step 2 and assuming that access rights allowed the requestor to subscribe to the resource, the NSCL asserts the identity for XDMS purposes.
- In step 3 the NSCL identifies the XDM subscription document (XCAP URI) using procedure (clause F.1.3) in that regard.
- In step 4 the NSCL issues the appropriate XDMS request to fetch the XDM document located in step 3.
- In step 5 XDMS returns an HTTP 200 OK response that includes the requested document.
- In step 6 the NSCL identifies the XCAP URI for the M2M target resource for subscription.
- In step 7 the NSCL issues a SIP SUBSCRIBE to XDMS for the target resource.
- In step 8 a 200 SIP OK is received.
- In step 9 the NSCL receives a SIP NOTIFY from XDMS that includes an XDM document for the initial state of the resource.
- In step 10 the NSCL issues a SIP 200 OK response back to XDMS.
- In step11 the NSCL updates the XDM subscription document fetched in step 4.
- In step 12 the NSCL returns the updated back to XMDS using XDMS procedures in that regard.
- In step 13 XDMS returns an HTTP 200 OK response back to M2M NSCL.
- In step 14 the NSCL maintains a state for the SIP subscription and a binding with the M2M subscription. This state can be saved within M2M NSCL or saved in XDMS in a proprietary fashion.
- In step 15 the NSCL returns a response to the incoming subscription request to report the outcome.

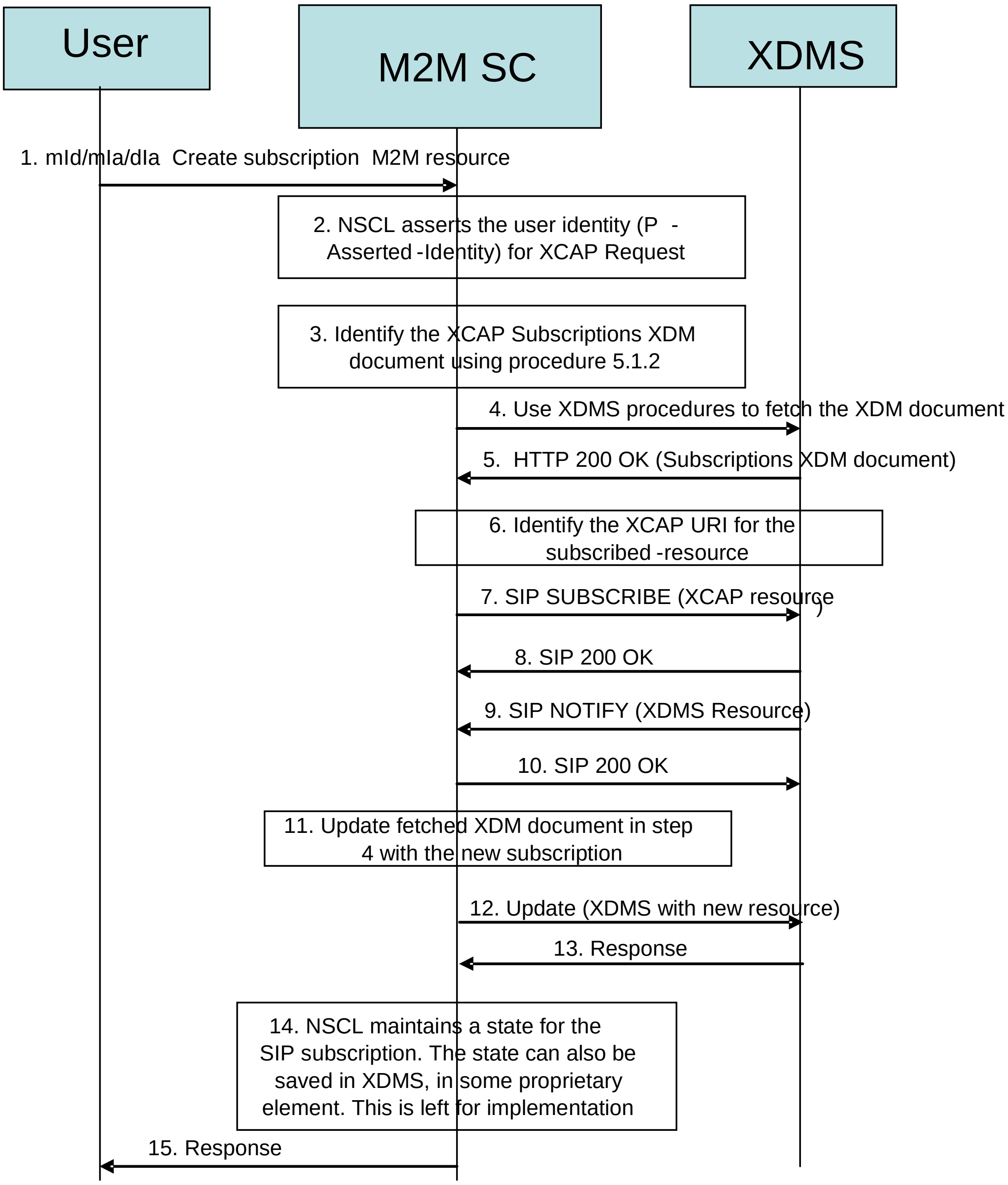

User
M2M SC
XDMS
1. mId/mIa/dIa Create subscription M2M resource
2. NSCL asserts the user identity (P - Asserted -Identity) for XCAP Request
3. Identify the XCAP Subscriptions XDM document using procedure 5.1.2
4. Use XDMS procedures to fetch the XDM document
5. HTTP 200 OK (Subscriptions XDM document)
6. Identify the XCAP URI for the subscribed -resource
7. SIP SUBSCRIBE (XCAP resource)
8. SIP 200 OK
9. SIP NOTIFY (XDMS Resource)
10. SIP 200 OK
11. Update fetched XDM document in step 4 with the new subscription
12. Update (XDMS with new resource)
13. Response
14. NSCL maintains a state for the SIP subscription. The state can also be saved in XDMS, in some proprietary element. This is left for implementation
15. Response

Figure F.60: Subscription Creation

F.3.2 Terminating a subscription

Figure F.61 shows a call flow for the procedure to be performed by the M2M NSCL for terminating a subscription.

The following is a brief description of the steps in the call flow:

- In step 1 the NSCL receives a request to delete an existing subscription to an M2M resource.
- In step 2 and assuming that access rights allowed the requestor to delete the subscription, the NSCL asserts the identity for XDMS purposes.
- In step 3 the NSCL identifies the XDM subscription document (XCAP URI) using procedure (clause F.1.3) in that regard.
- In step 4 the NSCL issues the appropriate XDMS request to fetch the XDM document located in step 3.
- In step 5, XDMS returns an HTTP 200 OK response that includes the requested document.
- In step 6 the NSCL identifies the XCAP URI for the M2M target resource for subscription deletion.
- In step 7, the NSCL locates the SIP state for the subscription matching the XCAP URI of step 6.
- In step 8 the NSCL issues a SIP SUBSCRIBE to XDMS for the target resource to terminate the subscription (expiry time =0).
- In step 9 a 200 SIP OK is received.
- In step 10 the NSCL updates the XDM subscription document fetched in step 4.
- In step 11 the NSCL returns the updated XDM document back to XMDS using XDMS procedures in that regard.
- In step 12 XDMS returns an HTTP 200 OK response back to M2M NSCL.
- In step 13 the NSCL removes the SIP state for the terminated subscription.
- In step 14, the NSCL returns a response to the incoming subscription request originator to report the outcome.

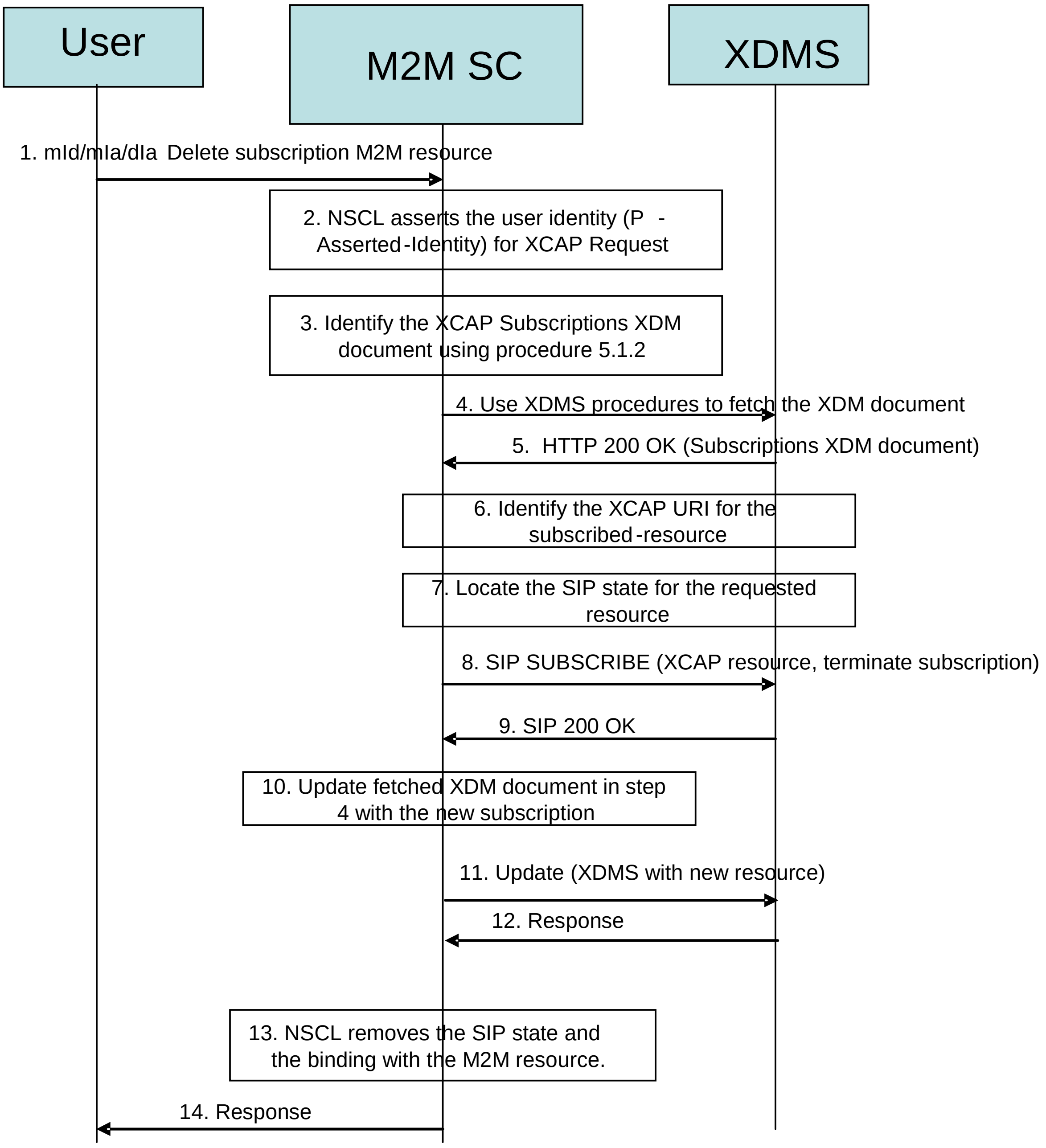

User
M2M SC
XDMS
1. mId/mIa/dIa Delete subscription M2M resource
2. NSCL asserts the user identity (P - Asserted-Identity) for XCAP Request
3. Identify the XCAP Subscriptions XDM document using procedure 5.1.2
4. Use XDMS procedures to fetch the XDM document
5. HTTP 200 OK (Subscriptions XDM document)
6. Identify the XCAP URI for the subscribed-resource
7. Locate the SIP state for the requested resource
8. SIP SUBSCRIBE (XCAP resource, terminate subscription)
9. SIP 200 OK
10. Update fetched XDM document in step 4 with the new subscription
11. Update (XDMS with new resource)
12. Response
13. NSCL removes the SIP state and the binding with the M2M resource.
14. Response

Figure F.61: Subscription Termination

F.4 Interworking between Gateways and XDMS

The assumption for interworking between a gateway and an XDMS is that the M2M SP and the network access provider are the same. The M2M SP can be the owner of the gateway. In case the M2M SP is not the owner of the gateway, business agreement shall be in place to allow interworking between the gateway and XDMS.

Figure F.62 shows the interworking between a gateway and an XDMS. The gateway can perform all operations that are performed by an NSCL. The gateway acts as a proxy in this scenario.

As a pre-requisite for any interworking, the gateway shall establish a TLS with the NSCL. To that effect, the gateway shall acts as an application and uses Kma to establish the TLS for that purpose.

The application ID allocated for interworking shall be as follows:

- <SCL-ID>-XDMS-proxy>.

Where <SCL-ID> is replaced by the SCL-ID for the gateway.

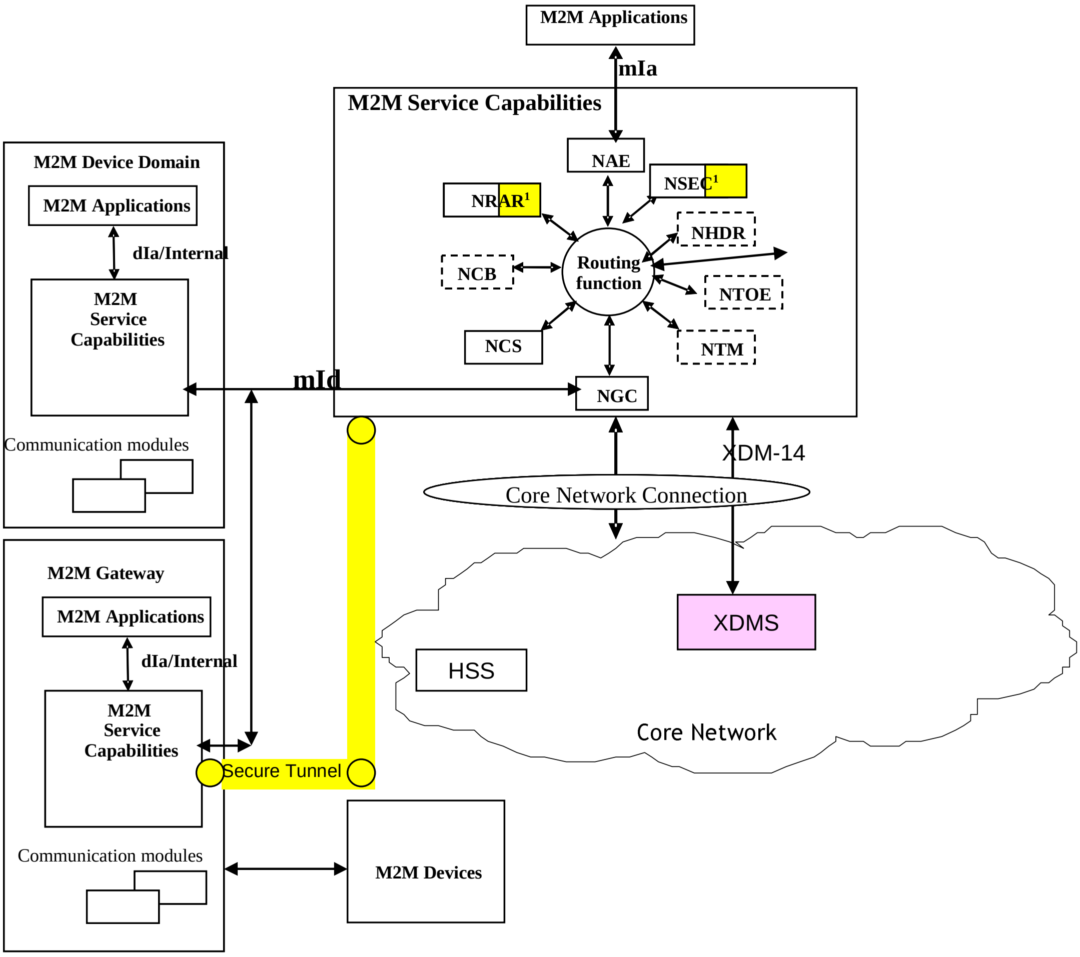


Figure F.62: Interworking Gateway - XDMS

F.5 Support for Managed Objects

F.5.1 Architectural View to Support M2M Management Objects

Figure F.63 show the proposed directory structure for supporting M2M management Objects.

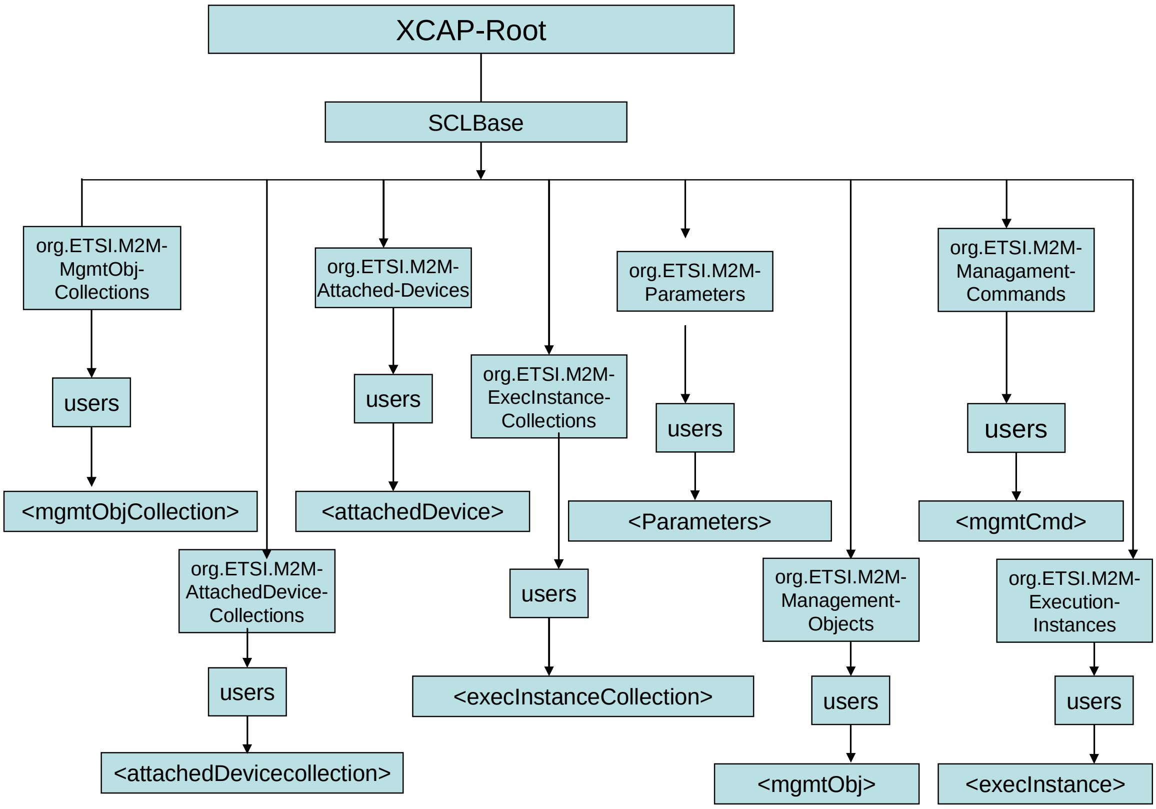

XCAP-Root
SCLBase
org.ETSI.M2M-MgmtObj-Collections
users
<mgmtObjCollection>
org.ETSI.M2M-Attached-Devices
users
<attachedDevice>
org.ETSI.M2M-ExecInstance-Collections
users
<execInstanceCollection>
org.ETSI.M2M-Parameters
users
<Parameters>
org.ETSI.M2M-Managament-Commands
users
<mgmtCmd>
org.ETSI.M2M-AttachedDevice-Collections
users
<attachedDevicecollection>
org.ETSI.M2M-Management-Objects
users
<mgmtObj>
org.ETSI.M2M-Execution-Instances
users
<execInstance>

Figure F.63: Overall Directory Structure to Support 2M2M Management Objects

Eight additional Application Usages are identified for managing M2M Management Objects resources. They are as follows:

- Management Object Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M management object collections resources.
- Attached Devices collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M attached devices collections resources.
- Execution Instances Collection Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M execution instance collections resources.
- Parameters Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M parameters resources.
- Attached Devices Application usage defined under the sclBase tree. This Application Usage handles the management of attached devices resources.
- Management Object Application usage defined under the sclBase tree. This Application Usage handles the management of Management Objects resources.
- Management Command Application usage defined under the sclBase tree. This Application Usage handles the management of M2M Management Commands resources.
- Execution Instance Application Usage defined under the sclBase tree. This Application Usage handles the management of M2M execution instances resources.

F.5.2 M2M Management Objects Application Usages

This clause lists the additional application usages that are required to support M2M management objects.

F.5.2.1 Management of MgmtObj Collection resources

F.5.2.1.1 MgmtObjs Collection Application Usage

The MgmtObjs collection Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read a collection of MgmtObj resources.

Every MgmtObj Collection resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the MgmtObj Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-MgmtObj-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-MgmtObj-Collections".

XML Schema

The M2M mgmtObjs Collection XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in mgmtObjs.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as described in table F.74.
- MgmtObj document, per XUI, under users' tree shall conform to the XML schema as defined in table F.74.
- Subscriptions document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an MgmtObj collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be five well-known documents. They are as follows:

- The well-known name of the first MgmtObj collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the MgmtObj Collection resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".

- The third well-known name of the MgmtObj Collection resource XDM document shall be "mgmtCmd". The document selector to access the "mgmtCmd" XDM document shall be "[auid]/users/[xui]/ mgmtCmd".
- The fourth well-known name of the MgmtObj Collection resource XDM document shall be "mgmtObj". The document selector to access the "mgmtObj" XDM document shall be "[auid]/users/[xui]/ mgmtObj".
- The fifth well-known name of the MgmtObj Collection resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/ subscriptions".

There is only a single instance for every document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

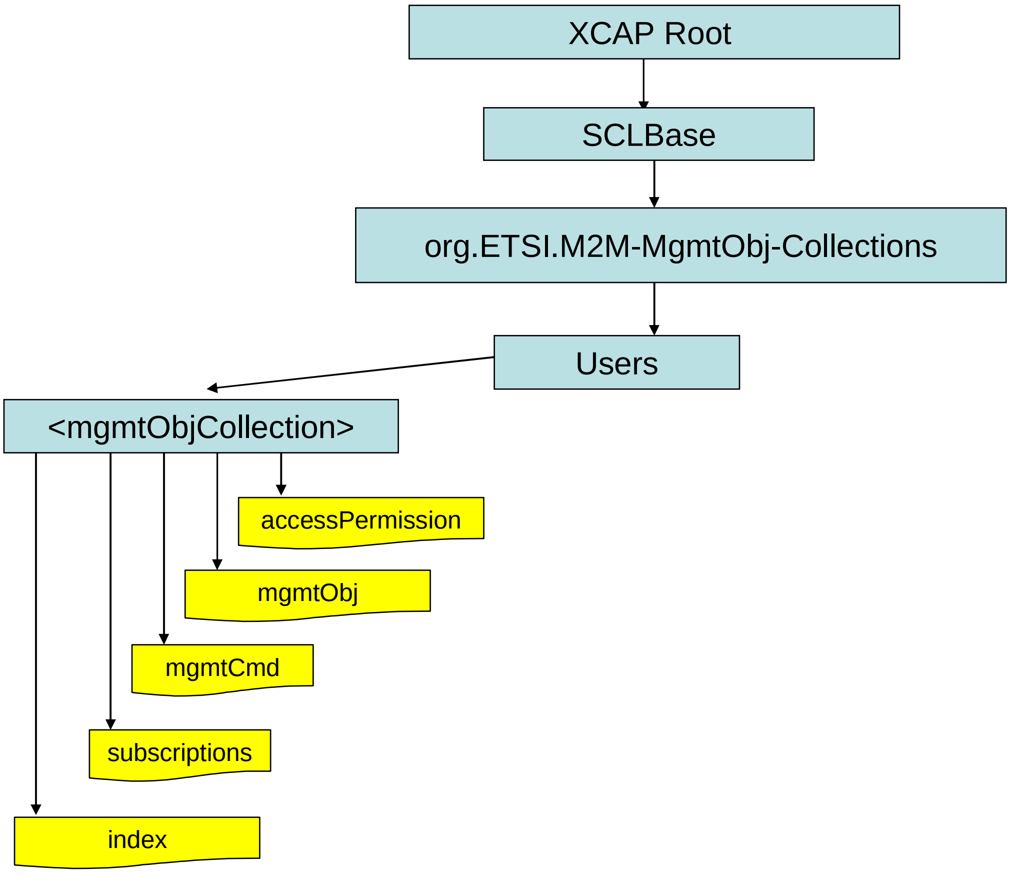


Figure F.64: Tree Structure for Management Object Collection Application Usage

F.5.2.1.2 Mapping between the XDMS XCAP <mgmtObjs> resources and M2M <mgmtObjs> resources

There are multiple locations within the M2M Resource tree structure where <mgmtObjs> resources are defined. As such, there are multiple cases for mapping between XDMS XCAP <mgmtObjs> and M2M <mgmtObjs> resources. However only one case will be described here.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/.

It is to be noted that a 1:1 mapping exists between M2M <mgmtObjs> resources and XDMS XCAP <mgmtObjs> resources.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.73: MgmtObj Collection M2M Resource URL Mapping <-> XCAP URL

M2M <mgmtObjs> Resources	XDMS XCAP <mgmtObjs> Resources
<sclBase>/scls/<scl>/mgmtObjs/ attribute (AccessRightID is mapped to a separate resource)	XCAPbase/XUI/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj>	XCAPbase/XUI/mgmtObj
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>	XCAPbase/XUI/mgmtCmd
<sclBase>/scls/<scl>/mgmtObjs/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls<scl>/mgmtObjs/attribute (AccessRightID)	XCAPbase/<XUI>/accessPermission

F.5.2.1.3 Information Mapping Between XDMS XCAP <mgmtObjs> resources and M2M <mgmtObjs> resources

Table F.74: mgmtObjs Collection XDMS Resource <-> M2M Resource

XDMS XCAP <mgmtObjs> Resource	M2M <mgmtObjs> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <mgmtObjs> resource except accessRightID
XCAPbase/XUI/mgmtObj	XDM document includes XCAP references to one or more mgmtObj document stored under the XUI tree of the mgmtObj AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users<mgmtObj>. The <mgmtObj> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no mgmtObj are defined in this collection
XCAPbase/XUI/mgmtCmd	XDM document includes XCAP references to one or more mgmtCmd document stored under the XUI tree of the mgmtCmd AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users<mgmtCmd>. The <mgmtCmd> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no mgmtCmd are defined in this collection
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.5.2.2 Management of AttachedDevice Collection resources

F.5.2.2.1 AttachedDevice Collection Application Usage

The AttachedDevice Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of AttachedDevices resources.

Each AttachedDevice Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the AttachedDevice Collection resource.

Note that there in only one attachedDevice Collection.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M- AttachedDevice-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M- AttachedDevice-Collections".

XML Schema

The AttachedDevice Collections Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in attachedDevices.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AttachedDevice document, per XUI, document under users' tree shall conform to the XML schema as defined in table F.76.
- Subscriptions document under users' tree, per XUI, shall be an empty document.
- accessPermission, per XUI, document under users' tree shall conform to an XCAP reference as defined in table F.76.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a AttachedDevice Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known AttachedDevice Collection XDM documents. They are as follows:

- The first well-known name of the AttachedDevice Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/ index".
- The second well-known name of the AttachedDevice Collection resource XDM document shall be "attachedDevice". The document selector to access the "attachedDevice" XDM document shall be "[auid]/users/[xui]/attachedDevice".
- The third well-known name of the AttachedDevice Collection resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The fourth well-known name of the AttachedDevice Collection resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/ subscriptions".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

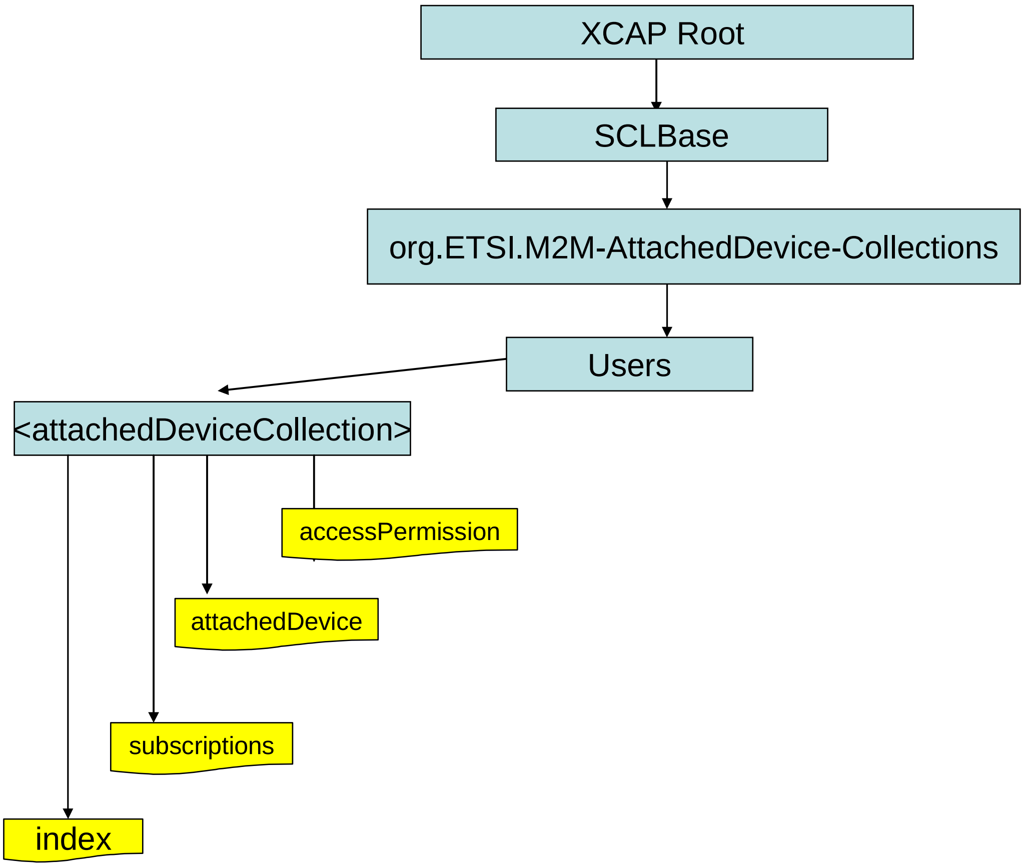


Figure F.65: Tree Structure for attachedDevice Collection Application Usage

F.5.2.2.2 Mapping between the XDMS XCAP <attachedDevices> resource URL and M2M < attachedDevices> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.75: AttachedDevice Collection M2M Resource URL Mapping <-> XCAP

M2M <attachedDevices> Resources	XDMS XCAP <attachedDevices> Resources
<sclBase>/scls/<scl>/attachedDevices/attribute	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/attachedDevices/<attachedDevice>	XCAPbase/XUI/attachedDevice
<sclBase>/scls/<scl>/attachedDevices/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/attachedDevices/attribute (AccessRightID)	XCAPbase/XUI/accessPermission

F.5.2.2.3 Information Mapping Between XDMS XCAP <attachedDevices> resource and M2M <attachedDevices> resource

Table F.76: AttachedDevice Collection XDMS Resource <-> M2M Resource

XDMS XCAP <attachedDevices> Resources	M2M <attachedDevices> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <attachedDevices> collection
XCAPbase/XUI/attachedDevice	XDM document includes XCAP references to one or more attachedDevice document stored under the XUI tree of the attachedDevice AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users<attachedDevice>. The <attachedDevice> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no attachedDevice is defined in this collection
XCAPbase/XUI/subscriptions	Empty Document
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here

F.5.2.3 Management of Attached Device resources

F.5.2.3.1 Attached Device Application Usage

The AttachedDevice Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read an AttachedDevice resource.

Every AttachedDevice resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the AttachedDevice resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Attached-Devices".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Attached-Devices".

XML Schema

The M2M attachedDevice XDM documents shall conform to the following XML schema:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in attachedDevice.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission document, per XUI, under users' tree shall conform to an XCAP reference as defined in table F.78.
- mgmtObjs document, per XUI, under users' tree shall be an empty document.
- Subscriptions document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M attachedDevice resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known documents for the AttachedDevice resource. They are as follows:

- The well-known name of the first AttachedDevice resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the AttachedDevice resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the AttachedDevice resource XDM document shall be "mgmtObjs". The document selector to access the "mgmtObjs" XDM document shall be "[auid]/users/[xui]/mgmtObjs".
- The fourth well-known name of the AttachedDevice resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/ subscriptions".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

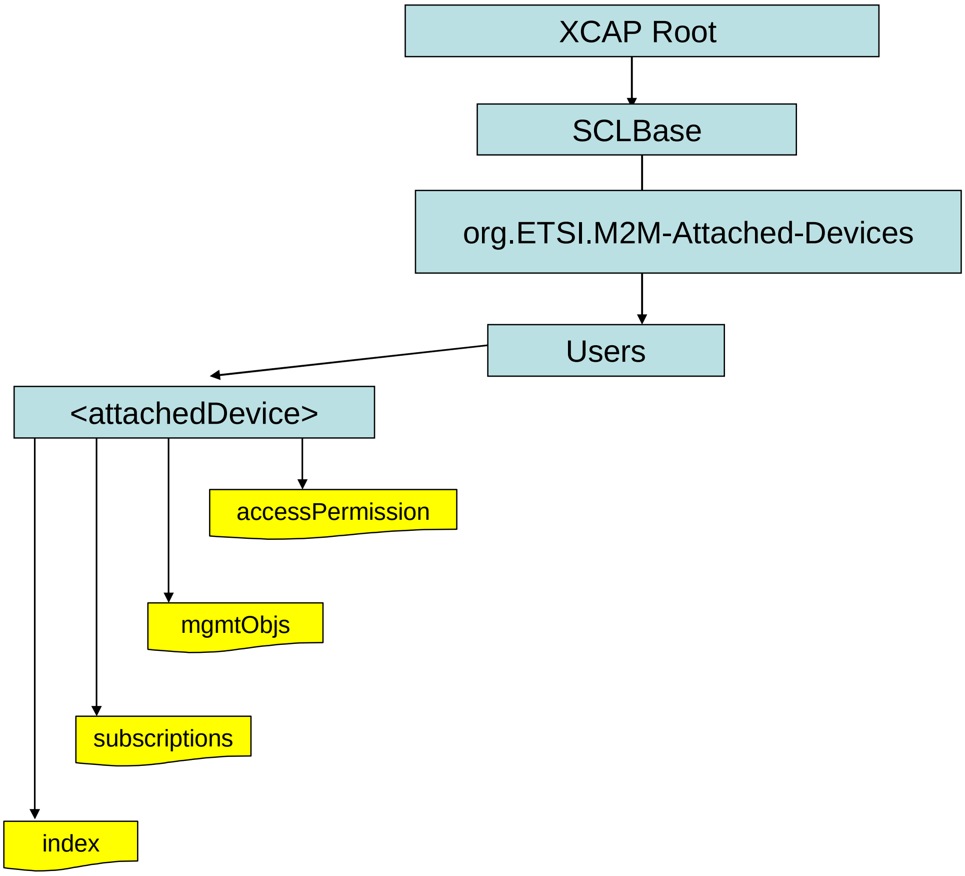


Figure F.66: Tree Structure for Attached Device Application Usage

F.5.2.3.2 Mapping between the XDMS XCAP <attachedDevice> resources and M2M <attachedDevice> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users/.

There is a 1:1 mapping between M2M <attachedDevice> resources and XDMS XCAP <attachedDevice> resources.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.77: AttachedDevice M2M Resource URL Mapping <-> XCAP URL

M2M <attachedDevice> Resources	XDMS XCAP <attachedDevice> Resources
<sclBase>/scls/<scl>/attachedDevices/<attacheDevice/attribute	XCAPbase/XUI/index
<sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/attribute (accessPermission)	XCAPbase/XUI/accessPermission
<sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/mgmtObjs	XCAPbase/XUI/mgntObjs
<sclBase>/scls/<scl>/attachedDevices/<attachedDevice>/subscriptions	XCAPbase/XUI/subscriptions

F.5.2.3.3 Information Mapping Between XDMS XCAP <attachedDevice> resources and M2M <attachedDevice> resources

Table F.78: AttachedDevices XDMS Resource <-> M2M Resource

XDMS XCAP <attachedDevice> Resource	M2M <attachedDevice> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <attachedDevice> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.
XCAPbase/XUI/mgntObjs	Empty document.
XCAPbase/XUI/subscriptions	Empty document.

F.5.2.4 Management of MgmtObj resources

F.5.2.4.1 MgmtObj Application Usage

The MgmtObj Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read a MgmtObj resource.

Every MgmtObj resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the MgmtObj resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Managament-Objects".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Management-Objects".

XML Schema

The M2M mgmtObj XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in mgmtObj.xsd, with the root element the sclBase, with the exception of the accessRightID attribute and all child references.
- AccessPermission, per XUI, document under users' tree shall conform to an XCAP reference as defined in table F.80.
- Parameters document, per XUI, under users' tree shall conform to the XML schema as described in table F.80.
- Subscriptions document, per XUI, under user' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a MgmtObj resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known documents for the MgmtObj resource. They are as follows:

- The well-known name of the first MgmtObj resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the MgmtObj resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the MgmtObj resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The fourth well-known name of the MgmtObj resource XDM document shall be "parameters". The document selector to access the "parameters" XDM document shall be "[auid]/users/[xui]/parameters".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

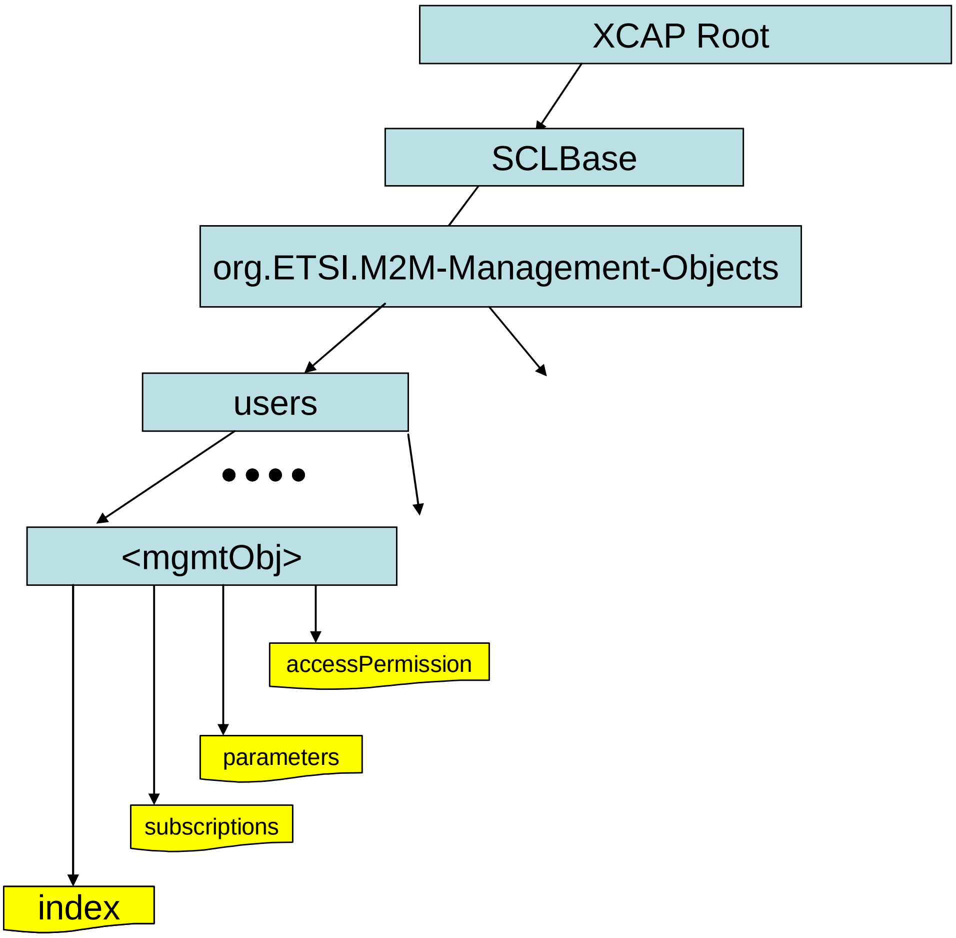


Figure F.67: Tree Structure for Management Object Application Usage

F.5.2.4.2 Mapping between the XDMS XCAP <mgmtObj> resources and M2M <mgmtObj> resources

There are multiple locations within the M2M Resource tree structure where <mgmtObj> resources are defined. As such, there multiple cases for mapping XDMS XCAP <mgmtObj> to M2M resources. However, only one case shall be considered here.

It is to be noted that a 1:1 mapping exists between M2M < mgmtObj> resources and XDMS XCAP < mgmtObj> resources.

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.79: mgmtObj M2M Resource URL Mapping <-> XCAP URL

M2M < mgmtObj> Resources	XDMS XCAP <mgmtObj> Resources
<sclBase>/scls/<scl>/mgmtObjs/<mgmntObj>/attribute	XCAPbase/XUI/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj>/parameters	XCAPbase/XUI/parameters
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj>/attribute (access Right)	XCAPbase/XUI/accessPermission

F.5.2.4.3 Information Mapping Between XDMS XCAP <mgmtObj> resources and M2M <mgmtObj> resources

Table F.80: mgmtObj XDMS Resource <-> M2M Resource

XDMS XCAP <mgmtObj> Resource	M2M <mgmtObj> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <mgmtObj> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.
XCAPbase/XUI/parameters	XDM document includes XCAP references to one or more parameters document stored under the XUI tree of the parameters AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users<parameters>. The <parameters> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no.
XCAPbase/XUI/subscriptions	Empty Document.

F.5.2.5 Management of MgmtCmd resources

F.5.2.5.1 MgmtCmd Application Usage

The MgmtCmd Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read a MgmtCmd resource.

Every MgmtCmd resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the MgmtCmd resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Management-Commands".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Management-Commands".

XML Schema

The M2M mgmtCmd XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in mgmtCmd.xsd, with the root element the sclBase, with the exception of the accessRightID attribute, attributes having separate XDM documents, and all child references.
- AccessPermission per XUI document under users' tree shall conform to an XCAP reference as described in table F.82.
- CmdType document, per XUI, under users' tree shall conform to the XML schema as described in table F.82.
- ExecReqArgs document, per XUI, under users' tree shall conform to the XML schema as described in table F.82.
- Execute document, per XUI, under users' tree shall conform to the XML schema as described in table F.82.
- Subscriptions document, per XUI, under user' tree shall be an empty document.
- execInstances document, per XUI, under user' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an MgmtCmd resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be seven well-known documents for the MgmtCmd resource. They are as follows:

- The well-known name of the first MgmtCmd resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the MgmtCmd resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the MgmtCmd resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".
- The fourth well-known name of the MgmtCmd resource XDM document shall be "cmdType". The document selector to access the "cmdType" XDM document shall be "[auid]/users/[xui]/cmdType".
- The fifth well-known name of the MgmtCmd resource XDM document shall be "execReqArgs". The document selector to access the "execReqArgs" XDM document shall be "[auid]/users/[xui]/execReqArgs".
- The six well-known name of the MgmtCmd resource XDM document shall be "execute". The document selector to access the "execute" XDM document shall be "[auid]/users/[xui]/execute".
- The seventh well-known name of the MgmtCmd resource XDM document shall be "execInstances". The document selector to access the "execInstances" XDM document shall be "[auid]/users/[xui]/execInstances".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

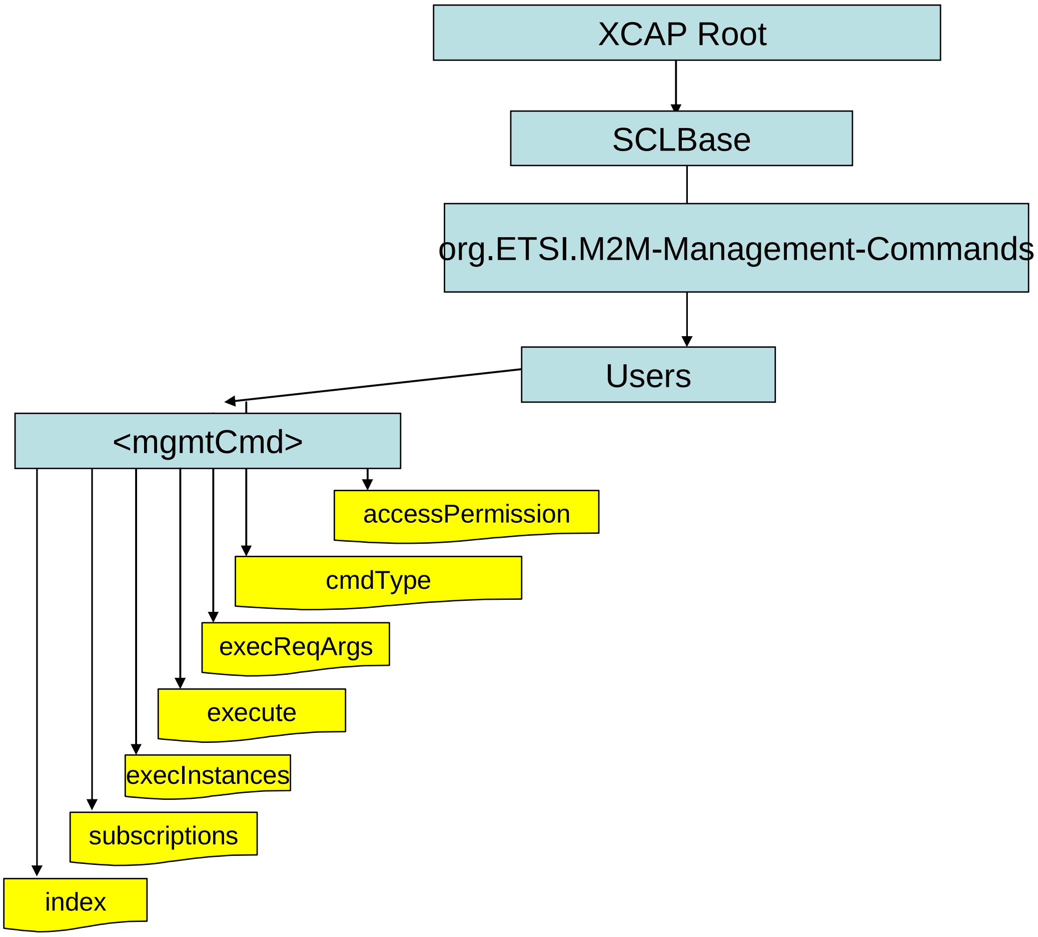


Figure F.68: Tree Structure for Management Command Application Usage

F.5.2.5.2 Mapping between the XDMS XCAP <mgmtCmd> resources and M2M <mgmtCmd> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/.

It is to be noted that a 1:1 mapping exists between M2M <mgmtCmd> resources and XDMS XCAP <mgmtCmd> resources.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.81: MgmtCmd M2M Resource URL Mapping <-> XCAP URL

M2M <mgmtCmd> Resources	XDMS XCAP <mgmtCmd> Resources
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/attribute	XCAPbase/XUI/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/cmdType	XCAPbase/XUI/cmdType
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execReqArgs	XCAPbase/XUI/execReqArgs
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execute	XCAPbase/XUI/execute
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances	XCAPbase/XUI/execInstances
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/attribute (accessRight)	XCAPbase/XUI/accessPermission

F.5.2.5.3 Information Mapping Between XDMS XCAP <mgmtCmd> resources and M2M <mgmtCmd> resources

Table F.82: mgmtCmd XDMS Resource <-> M2M Resource

XDMS XCAP <mgmtCmd> Resource	M2M <mgmtCmd> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the < mgmtCmd > resource (except the ones below).
XCAPbase/XUI/cmdType	XDM document conforms to the XML schema for cmdType attribute defined for the <mgmtCmd> resource.
XCAPbase/XUI/execReqArgs	**XDM document conforms to the XML schema for execReqArgs attribute defined for the <mgmtCmd> resource**.
XCAPbase/XUI/execute	XDM document conforms to the XML schema for execute attribute defined for the <mgmtCmd> resource.
XCAPbase/XUI/execInstances	Empty Document.
XCAPbase/XUI/subscriptions	Empty Document.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.

F.5.2.6 Management of ExecInstance Collection resources

F.5.2.6.1 ExecInstance Collection Application Usage

The ExecInstance Collection Application Usage allows an M2M entity to be able to create/modify/delete a collection of ExecInstance resources.

Each ExecInstance Collection resource that is created shall be allocated a unique identity that is under the users' tree beneath the AUID tree allocated to the ExecInstance Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-ExecInstance-Collections".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-ExecInstance-Collections".

XML Schema

The execInstance Collections Resource XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in execInstances.XSD, with the root element the sclBase, with the exception of the accessRighID attribute and all child references.

- accessPermission per XUI document under users' tree shall conform to an XCAP reference as defined in table F.84.
- execInstance document, per XUI, under users' tree shall conform to the XML schema as described in table F.84.
- Subscriptions document, per XUI, under users' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a ExecInstance Collection resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known ExecInstance Collection XDM documents. They are as follows:

- The first well-known name of the ExecInstance Collection resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the ExecInstance Collection XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the ExecInstance Collection XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/ subscriptions".
- The fourth well-known name of the ExecInstance Collection XDM document shall be "execInstance". The document selector to access the "execInstance" XDM document shall be "[auid]/users/[xui]/ "execInstance".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

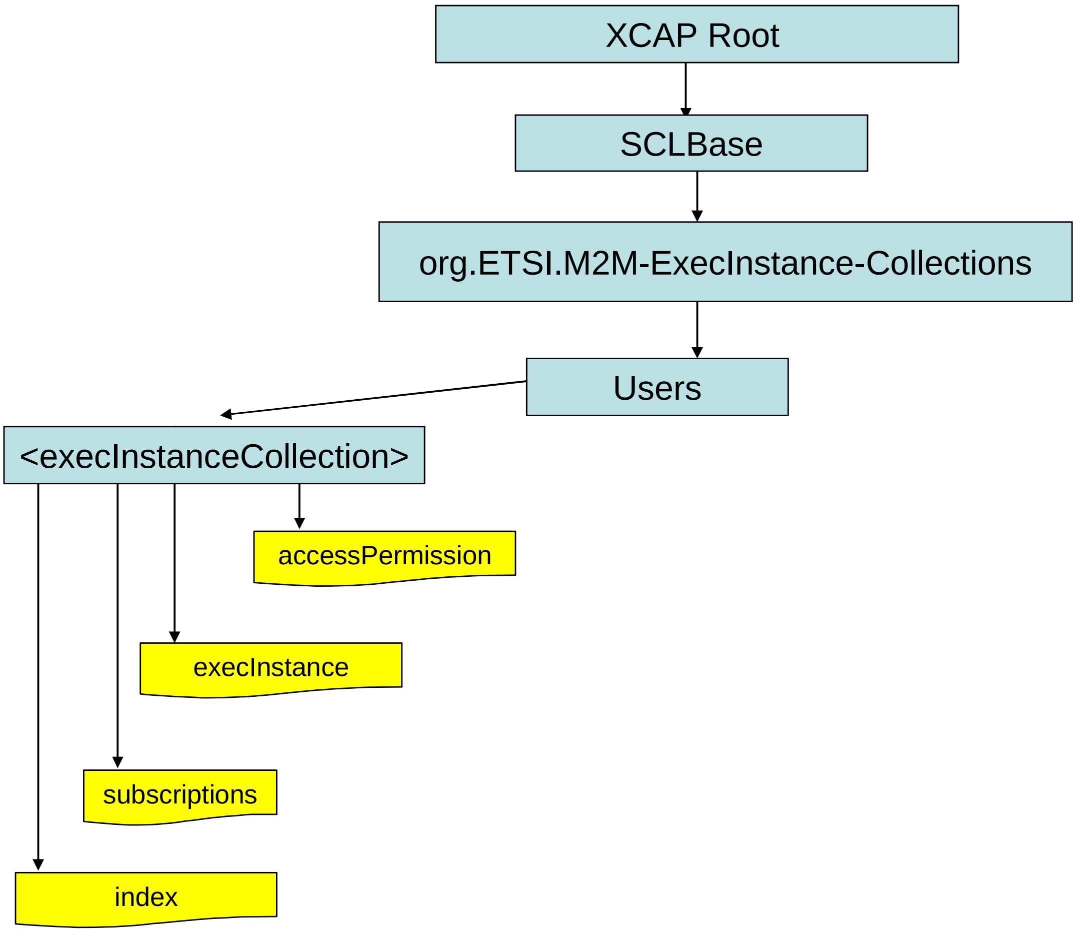


Figure F.69: Tree Structure for Execution Instance Collection Application Usage

F.5.2.6.2 Mapping between the XDMS XCAP <execInstances> resource URL and M2M <execInstances> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.83: ExecInstance Collection M2M Resource URL Mapping <-> XCAP

M2M <execInstances> Resources	XDMS XCAP <execInstances> Resources
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances /attribute	XCAPbase/<XUI>/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances /<execInstance>	XCAPbase/<XUI>/execInstance
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances /subscriptions	XCAPbase/<XUI>/subscriptions
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances /accessPermission	XCAPbase/<XUI>/accessPermission

F.5.2.6.3 Information Mapping Between XDMS XCAP <execInstances> resource and M2M <execInstances> resource

Table F.84: ExecInstance Collection XDMS Resource <-> M2M Resource

XDMS XCAP <execInstances> Resources	M2M < execInstances> Resources
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <execInstances> resource except accessRightID
XCAPbase/XUI/execInstance	XDM document includes XCAP references to one or more execInstance document stored under the XUI tree of the execInstance AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Exec-Instances/users<execInstance>. The <execInstance> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no execInstance is defined in this collection
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here
XCAPbase/<XUI>/subscriptions	Empty Document

F.5.2.7 Management of ExecInstance resources

F.5.2.7.1 ExecInstance Application Usage

The ExecInstance Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read M2M execInstance resources.

Every M2M execInstance resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the M2M ExecInstance resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Execution-Instances".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Execution-Instances".

XML Schema

The M2M executionsInstance XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in execInstance.xsd, with the root element the sclBase, with the exception of accessRighID attribute, attributes with a separate XDM document, and all child references.
- AccessPermission per XUI document under users' tree shall conform to an XCAP reference as defined in table F.86.
- Status document, per XUI, under users' tree shall conform to the XML schema as described in table F.86.
- Result document, per XUI, under users' tree shall conform to the XML schema as described in table F.86.
- Cancel document, per XUI, under users' tree shall conform to the XML schema as described in table F.86.
- Subscriptions document, per XUI, under user' tree shall be an empty document.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to a M2M ExecInstance resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be six well-known documents for the ExecInstances resource. They are as follows:

- The well-known name of the first ExecInstance resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the ExecInstance resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the ExecInstance resource XDM document shall be "status". The document selector to access the "status" XDM document shall be "[auid]/users/[xui]/status".
- The fourth well-known name of the ExecInstance resource XDM document shall be "result". The document selector to access the "result" XDM document shall be "[auid]/users/[xui]/result".
- The fifth well-known name of the ExecInstance resource XDM document shall be "cancel". The document selector to access the "cancel" XDM document shall be "[auid]/users/[xui]/cancel".
- The six well-known name of the ExecInstance resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

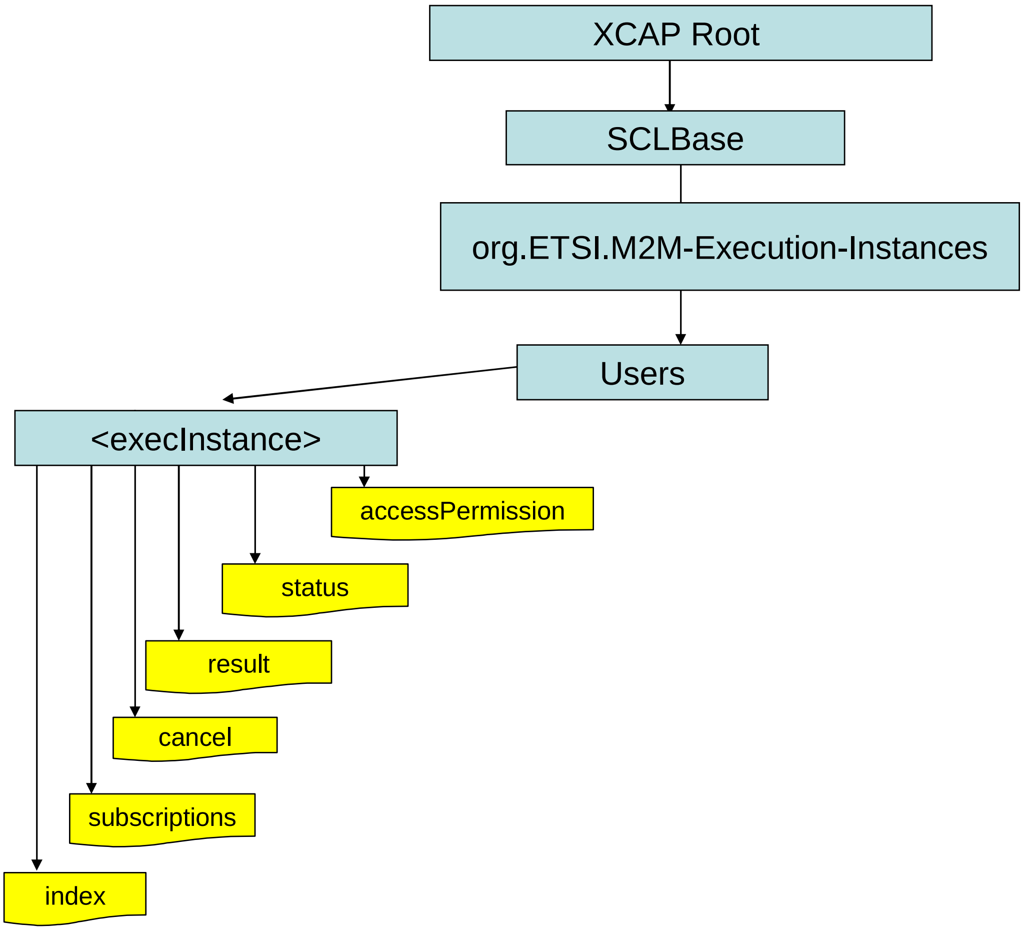


Figure F.70: Tree Structure for Execution Instance Application Usage

F.5.2.7.2 Mapping between the XDMS XCAP <execInstance> resources and M2M <execInstance> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/.

It is to be noted that a1:1 mapping exists between M2M <execInstance> resources and XDMS XCAP <execInstance> resources.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.85: ExecInstance M2M Resource URL Mapping <-> XCAP URL

M2M <execInstance> Resources	XDMS XCAP <execInstance> Resources
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>/attribute	XCAPbase/XUI/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>/status	XCAPbase/XUI/status
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>/result	XCAPbase/XUI/result
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>cancel	XCAPbase/XUI/cancel
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/mgmtObjs/<mgmtCmd>/execInstances/<execInstance>/accessPermission	XCAPbase/XUI/accessPermission

F.5.2.7.3 Information Mapping Between XDMS XCAP <execInstance> resources and M2M <execInstance> resources

Table F.86: ExecInstance XDMS Resource <-> M2M Resource

XDMS XCAP <execInstance> Resource	M2M <execInstance> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <execInstance> resource (except the ones below).
XCAPbase/XUI/status	XDM document conforms to the XML schema for the status attribute defined for the <execInstance> resource.
XCAPbase/XUI/result	XDM document conforms to the XML schema for the result attribute defined for the <execInstance> resource.
XCAPbase/XUI/cancel	XDM document conforms to the XML schema for the cancel attribute defined for the <execInstance> resource.
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here.
XCAPbase/XUI/subscriptions	Empty XDM document.

F.5.2.8 Management of Parameters resources

The Parameter Application Usage allows an M2M entity to be able to Create/Modify/Delete/Read a Parameter resource.

Every Parameter resource that is created shall be allocated a unique identity and shall be located under the users' tree located beneath the AUID tree allocated to the Parameter Collection resource.

Application Unique ID

The AUID SHALL be "org.ETSI.M2M-Parameters".

Default Namespace

The default namespace SHALL be "urn:etsi:xml:xdm:M2M-Parameters".

XML Schema

The M2M parameters XDM documents shall conform to the following XML schemas:

- Index document, per XUI, under users' tree shall conform to the XSD defined in the present document, annex B, specified in parameters.xsd, with the root element the sclBase, with the exception of accessRightID attribute and all child references.
- AccessPermission per XUI document under users' tree shall conform to an XCAP reference as defined in table F.88.
- Subscriptions document, per XUI, under users' tree shall be an empty document.
- Parameters document, per XUI, under users' tree shall conform to the XML schema as described in table F.88.

Additional Constraints

In addition to conforming to the XML schema, the XDMS shall ensure that every identity allocated to an M2M Parameter resource is unique.

The request originator shall be extracted from the X-3GPP-Asserted-Identity HTTP header and shall be used by the XDMS for validating access rights for the requested operation before the operation is accepted.

Data Semantics

The semantics for the data is defined in the present document, annex B.

Naming Conventions

There shall be four well-known documents for the Parameter resource. They are as follows:

- The well-known name of the first Parameter resource XDM document shall be "index". The document selector to access the "index" XDM document shall be "[auid]/users/[xui]/index".
- The second well-known name of the Parameter resource XDM document shall be "accessPermission". The document selector to access the "accessPermission" XDM document shall be "[auid]/users/[xui]/accessPermission".
- The third well-known name of the M2M Parameter resource XDM document shall be "parameters". The document selector to access the "parameters" XDM document shall be "[auid]/users/[xui]/parameters".
- The fourth well-known name of the M2M Parameter resource XDM document shall be "subscriptions". The document selector to access the "subscriptions" XDM document shall be "[auid]/users/[xui]/subscriptions".

There is only a single instance for the present document.

Authorization Policies

The XDMS server shall authorize requests for all operations on XDM documents located under the XUI tree using the accessPermission document under the same XUI tree for that purpose.

Global Document

There are no global documents under the global tree.

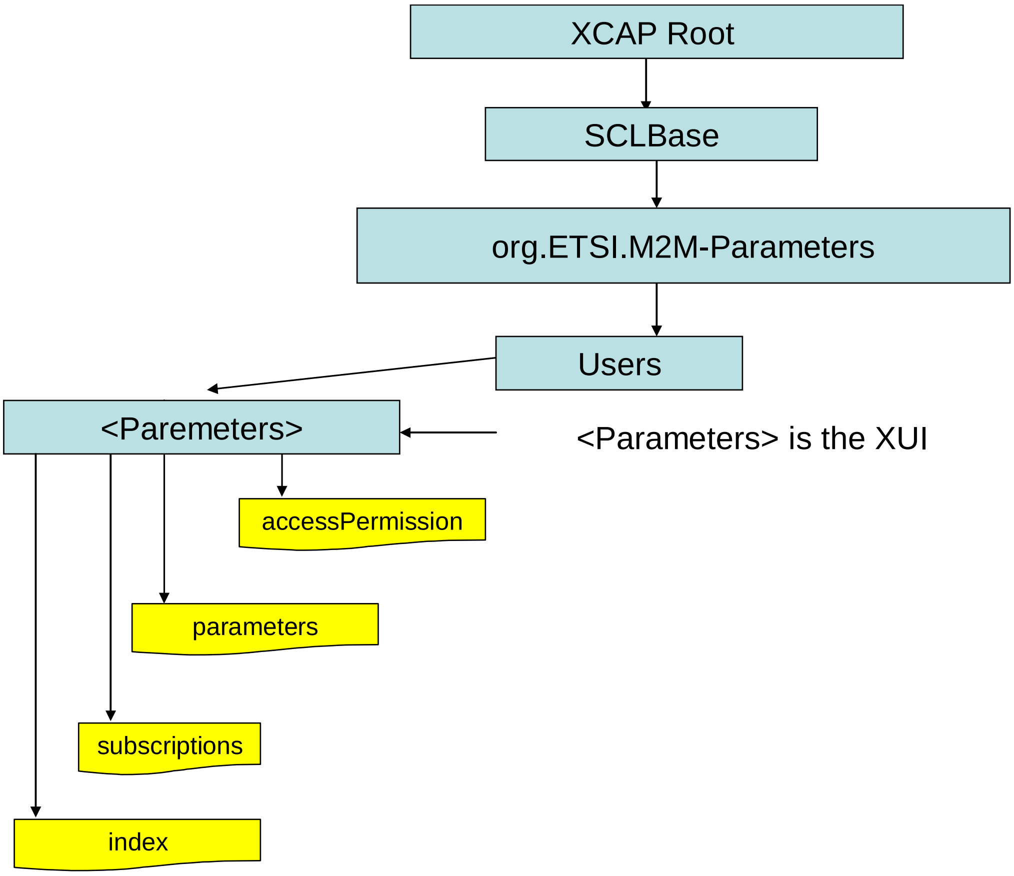


Figure F.71: Tree Structure for Parameters Application Usage

F.5.2.8.1 Mapping between the XDMS XCAP <parameters> resources and M2M <parameters> resources

The XCAP base for this application usage shall be XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/.

It is to be noted that a 1:1 mapping exists between M2M <parameters> resources and XDMS XCAP <parameters> resources.

Note that the mapping between the M2M URI and the XUI in the XDMS XCAP resource is governed by clause F.1.3.1.

Table F.87: Parameters M2M Resource URL Mapping <-> XCAP URL

M2M <parameters> Resources	XDMS XCAP <parameters> Resources
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj/attribute	XCAPbase/XUI/index
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj/parameters	XCAPbase/XUI/parameters
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj/subscriptions	XCAPbase/XUI/subscriptions
<sclBase>/scls/<scl>/mgmtObjs/<mgmtObj>/accessPermission	XCAPbase/XUI/accessPermission

F.5.2.8.2 Information Mapping Between XDMS XCAP <parameters> resources and M2M <parameters> resources

Table F.88: Parameters XDMS Resource <-> M2M Resource

XDMS XCAP <parameters> Resource	M2M <parameters> Resource
XCAPbase/XUI/index	XDM document conforms to the XML schema for all attributes defined for the <parameters> resource
XCAPbase/XUI/parameters	XDM document includes XCAP references to one or more parameters document stored under the XUI tree of the Parameters AUID. Each XCAP reference will have the following URL: XCAPRoot/<sclBase>/org.ETSI.M2M-Patameters/users<parameters>. The <parameters> document shall be created and available first before it can be referenced. Note that the XDM document may include no XCAP references at all as well, if no parameterCollection are defined in this collection
XCAPbase/XUI/accessPermission	XDM document includes an XCAP reference to an <accessRight> document stored under the XUI tree of the Access Right AUID. The XCAP reference will have the following URL: XCAPRoot//<sclBase>/org.ETSI.M2M-Access-Rights/users<accessRight>. The <accessRight> document shall be first created or available so its reference can be stored here
XCAPbase/XUI/subscriptions	Empty XDM document

F.5.3 Impacts on existing Application Usage due to M2M Management Objects

Three already defined application usages are also impacted due to the support of M2M Management Objects. This section describes the impacts on existing application usages.

F.5.3.1 Enhanced Registered SCL Application Usage

This section details the addition specification for the Registered SCL application usage to support M2M management objects.

XML Schema

The Registered SCL XDM documents shall conform to the following XML schemas for the additional XDM documents required to support M2M management Objects:

- mgmtObjs document, per XUI, under users' tree shall be an empty document.
- attachedDevices document, per XUI, under users' tree shall be an empty document.

Naming Conventions

There shall be two additional well-known documents. They are as follows:

- The first additional well-known document shall be "mgmtObjs". The document selector to access the "mgmtObjs" XDM document shall be "[auid]/users/[xui]/mgmtObjs".
- The second additional well-known document shall be "attachedDevices". The document selector to access the "attachedDevices" XDM document shall be "[auid]/users/[xui]/attachedDevices".

There is only a single instance for each additional document.

F.5.3.1.1 Mapping between XDMS XCAP <scl> resources and M2M <scl> resources

Table F.89: Registered SCL M2M Resource URL Mapping <-> XCAP URL for supporting M2M Management Objects

M2M <scl> Resources	XDMS XCAP <scl> Resources
<sclBase>/scls/<scl>/mgmbObjs	XCAPbase/<XUI>/mgmtObjs
<sclBase>/scls/<scl>/attachedDevices	XCAPbase/<XUI>/attachedDevices

F.5.3.1.2 Information Mapping between the XDMS XCAP <scl> resources and M2M <scl> resources

Table F.90: Registered SCL XDMS Resource <-> M2M Resource for supporting Management Objects

XDMS XCAP <scl> Resource	M2M <scl> Resource
XCAPbase/<XUI>/mgmtObjs	Empty Document
XCAPbase/<XUI>/attachedDevices	Empty Document

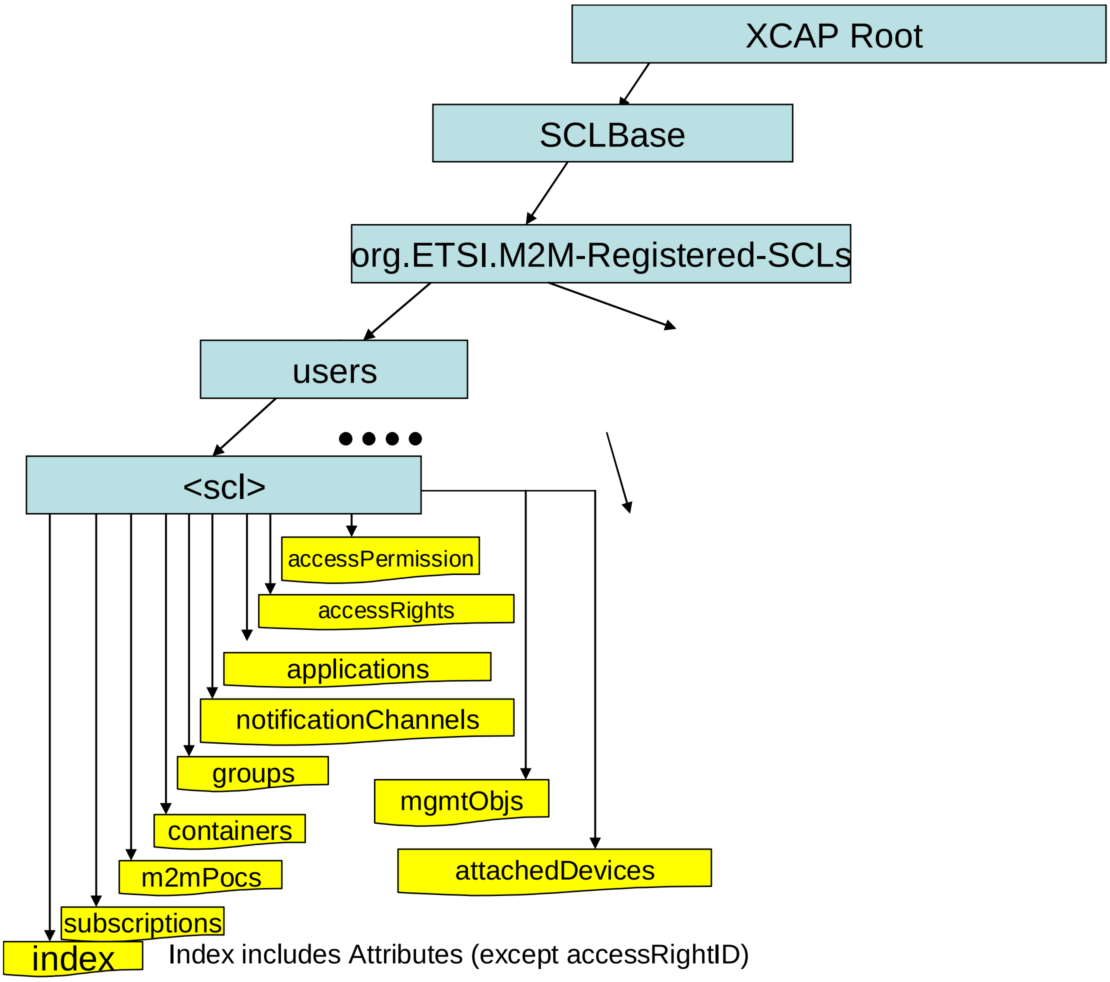


Figure F.72: Enhanced Registered SCL Application Usage to support M2M Management Objects

F.5.3.2 Enhanced Application Collection Application Usage

This clause details the additional specification for the Application Collection application usage to support M2M management objects.

XML Schema

The Applications Collection XDM documents shall conform to the following XML schema for the following additional XDM documents required to support M2M Management Objects:

- mgmtObjs document, per XUI, under users' tree shall be an empty document.

Naming Conventions

There shall be one additional well-known document. It is s follows:

- The first additional well-known document shall be "mgmtObjs". The document selector to access the mgmtObjs XDM document shall be "[auid]/users/[xui]/mgmtObjs".

There is only a single instance for this additional document.

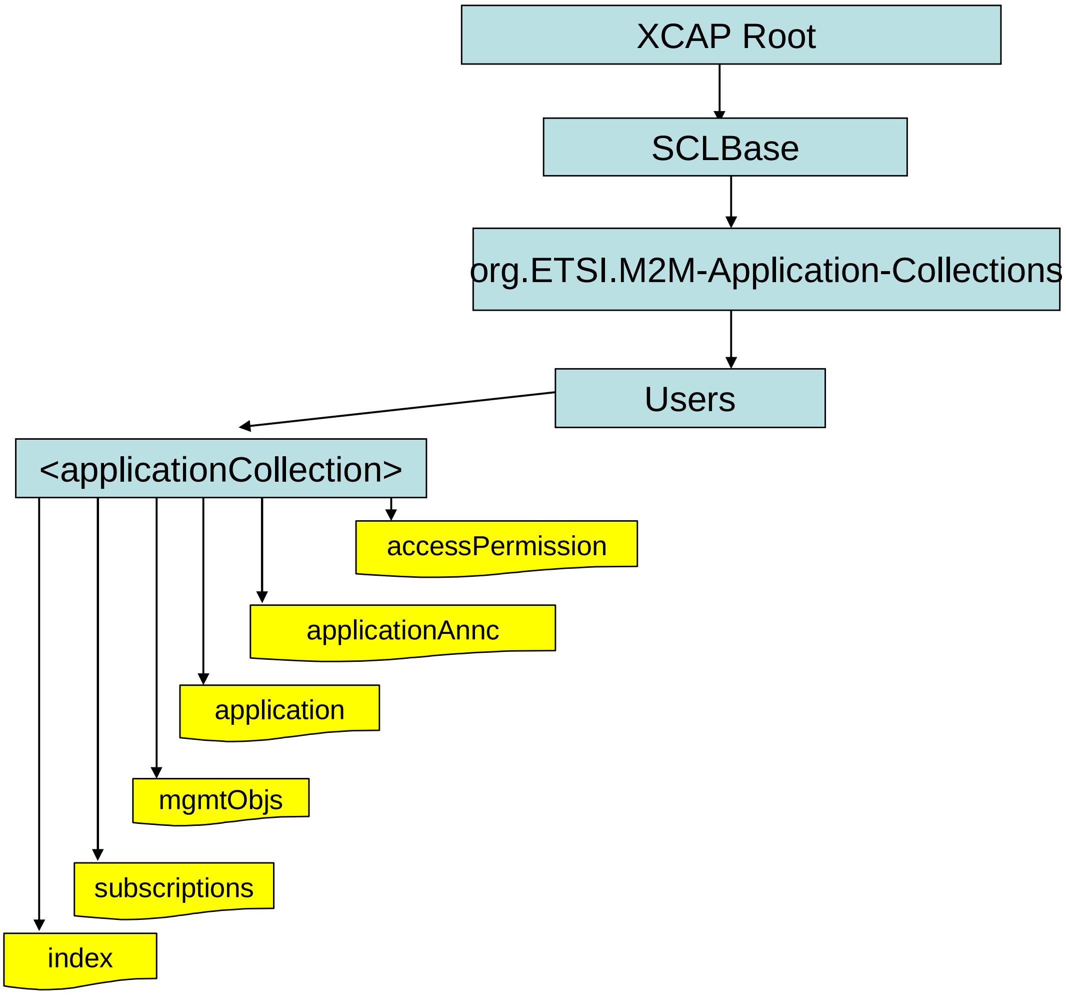


Figure F.73: Tree Structure for Enhanced Application Collection Application Usage

F.5.3.2.1 Mapping between XDMS XCAP <applications> resources and M2M <applications> resources

Table F.91: Application Collection M2M Resource URL Mapping <-> XCAP URL for supporting M2M Management Objects

M2M <applications> Resources	XDMS XCAP <applications> Resources
<sclBase>/scls/<scl>/applications/mgmbObjs	XCAPbase/<XUI>/mgmtObjs

F.5.3.2.2 Information Mapping between the XDMS XCAP <applications> resources and M2M <applications> resources

Table F.92: Application Collection XDMS Resource <-> M2M Resource for supporting Management Objects

XDMS XCAP <applications> Resource	M2M <applications> Resource
XCAPbase/<XUI>/mgmtObjs	Empty Document

F.5.3.3 SCL Collection Application Usage

This clause details the additional specification for the SCL Collection application usage to support the M2M management objects.

XML Schema

The SCL Collection XDM documents shall conform to the following XML schemas for the following additional XDM documents required to support M2M Management Objects:

- mgmtObjs document per XUI under users' tree shall be an empty document.

Naming Conventions

There shall be one additional well-known document:

- The first additional well-known document shall be "mgmtObjs". The document selector to access the mgmtObjs XDM document shall be "[auid]/users/[xui]/mgmtObjs".

There is only a single instance for this additional document.

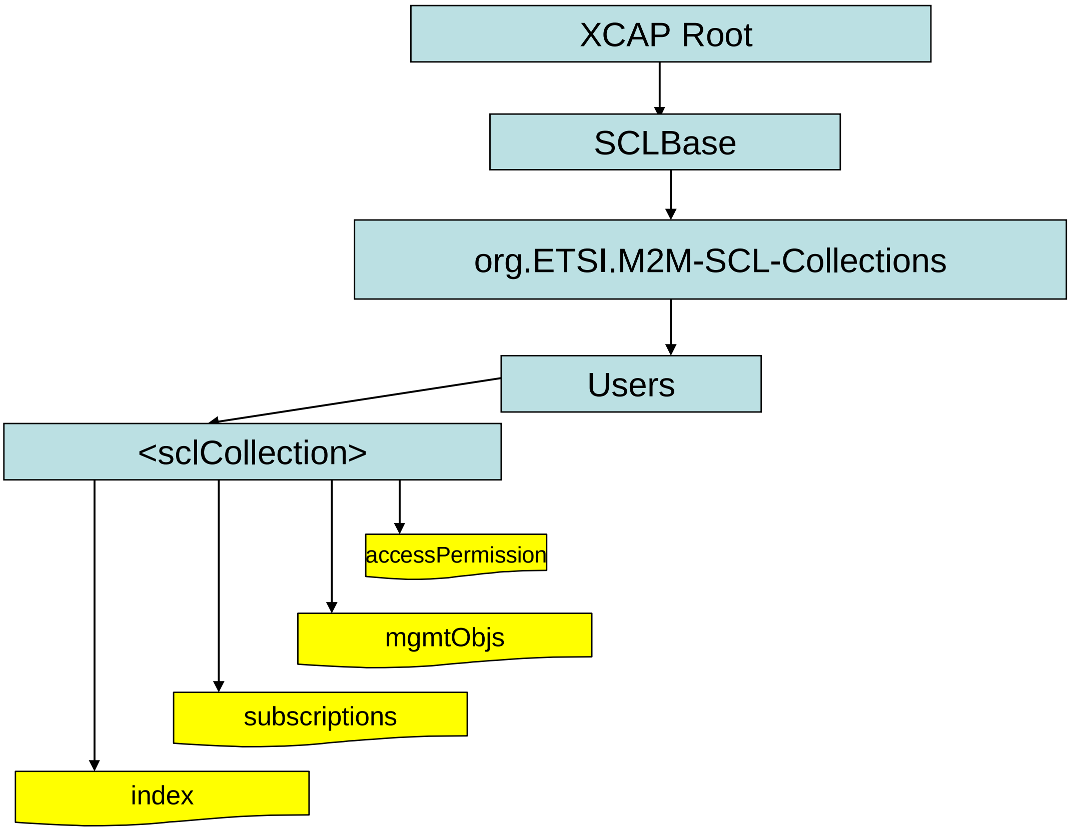


Figure F.74: Tree Structure for Enhanced SCL Collection Application Usage

F.5.3.3.1 Mapping between XDMS XCAP <scls> resources and M2M <scls> resources

Table F.93: SCL Collection M2M Resource URL Mapping <-> XCAP URL for supporting M2M Management Objects

M2M <scls> Resources	XDMS XCAP <scls> Resources
<sclBase>/scls/mgmbObjs	XCAPbase/<XUI>/mgmtObjs

F.5.3.3.2 Information Mapping between the XDMS XCAP <scls> resources and M2M <scls> resources

Table F.94: SCL Collection XDMS Resource <-> M2M Resource for supporting Management Objects

XDMS XCAP <scls> Resource	M2M <scls> Resource
XCAPbase/<XUI>/mgmtObjs	Empty Document

F.6 Network Procedures in Support of M2M Management Objects

This clause provides a list of the basic procedures that shall be performed by the NSCL in the Network domain to manage the different M2M resources stored in XDMS that are associated with M2M Management Objects.

F.6.1 Resource Creation In support of M2M Managed Objects

F.6.1.1 Enhancement to the Creation of an scl Resource

Figure F.75 show the procedures to be performed by the NSCL in the Network domain to create a registered SCL XDM resource.

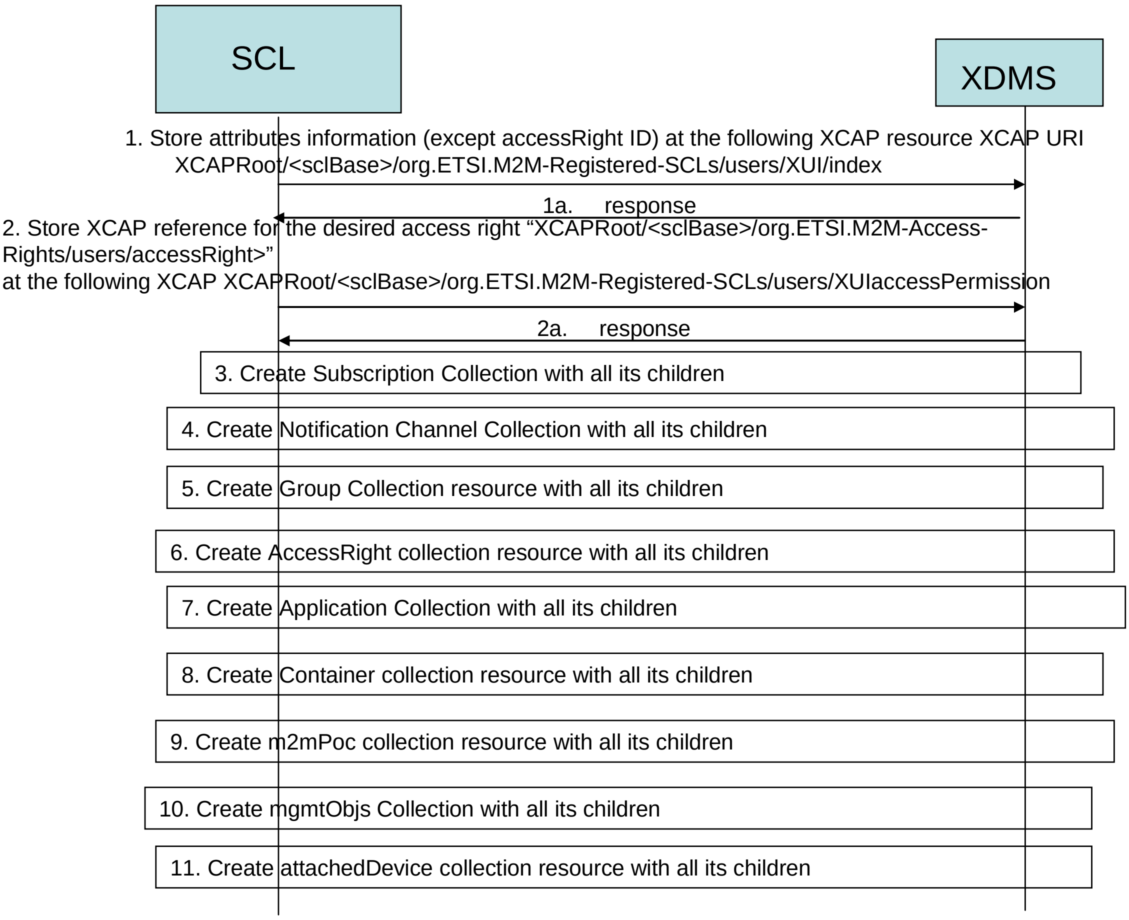

SCL
XDMS
1. Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUI/index
1a. response
2. Store XCAP reference for the desired access right "XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/accessRight>"
at the following XCAP XCAPRoot/<sclBase>/org.ETSI.M2M-Registered-SCLs/users/XUIaccessPermission
2a. response
3. Create Subscription Collection with all its children
4. Create Notification Channel Collection with all its children
5. Create Group Collection resource with all its children
6. Create AccessRight collection resource with all its children
7. Create Application Collection with all its children
8. Create Container collection resource with all its children
9. Create m2mPoc collection resource with all its children
10. Create mgmtObjs Collection with all its children
11. Create attachedDevice collection resource with all its children

Figure F.75: Enhanced Registered SCL Resource creation process

The following is a brief description of the steps in the call flow:

- Steps 1-9 are already described in clause F.2.1.1.
- In step 10, Network Domain (NSCL) creates a mgmtObjs collection resource using procedure (clause F.6.1.6) for that matter.
- In step 11, Network Domain (NSCL) creates an attachedDevice collection resource using procedure (clause F.6.1.4) for that matter.

F.6.1.2 Enhancement to the Creation of an Application Collection

Figure F.76 shows the procedure to be performed by the NSCL in the Network domain to create an application collection XDM resource.

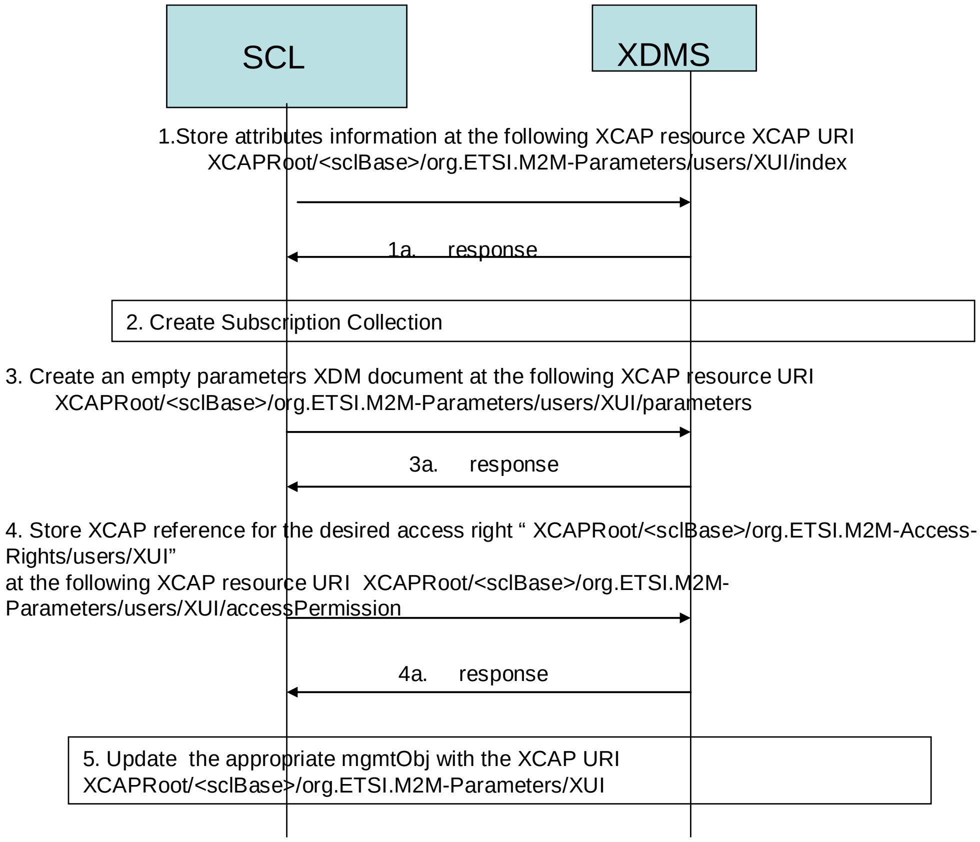

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty parameters XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/parameters
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/accessPermission
4a. response
5. Update the appropriate mgmtObj with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/XUI

Figure F.76: Application Collection Resource creation process

The following is a brief description of the steps in the call flow:

- Steps 1-5 are already explained in clause F.2.16 and will not be repeated here.
- In step 6, Network Domain (NSCL) creates a mgmtObjs collection resource using procedure (clause F.6.1.6) for that matter.

F.6.1.3 Enhancements to the Creation of SCL Collection

Figure F.77 shows the procedure to be performed by the Network domain (NSCL) to create an scl collection XDM resource.

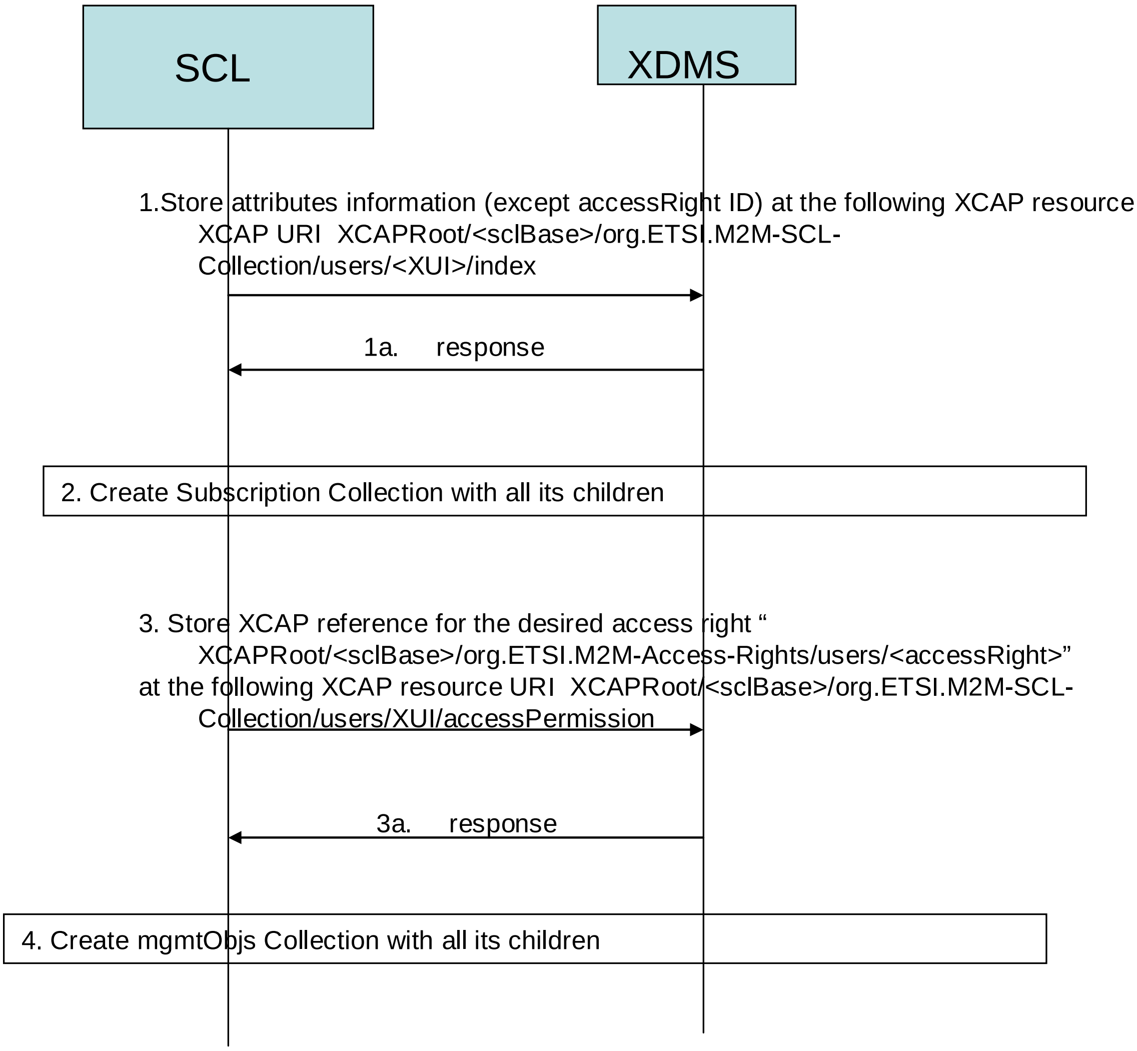

SCL
XDMS
1.Store attributes information (except accessRight ID) at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/<XUI>/index
1a. response
2. Create Subscription Collection with all its children
3. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/<accessRight>" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-SCL-Collection/users/XUI/accessPermission
3a. response
4. Create mgmtObjs Collection with all its children

Figure F.77: SCL Collection Resource creation process

The following is a brief description of the steps in the call flow:

- Steps 1 - 3 are already described in clause F.2.17.
- In step 4, Network Domain (NSCL) creates a mgmtObjs collection resource using procedure (clause F.6.1.6) for that matter.

F.6.1.4 Creation of Attached Devices Collection

Figure F.78 shows the procedure to be performed by the NSCL in the Network domain to create an attached device collection XDM resource.

In this case, the Network domain (NSCL) creates a new tree representing the attached device collection resource under the users' tree as per the Attached Device Collection application usage. The Network domain (NSCL) creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/) for the attachedDevice collection applying procedure (clause F.1.3.1.1). This results in the XUI being the same as the parent resource for the collection.

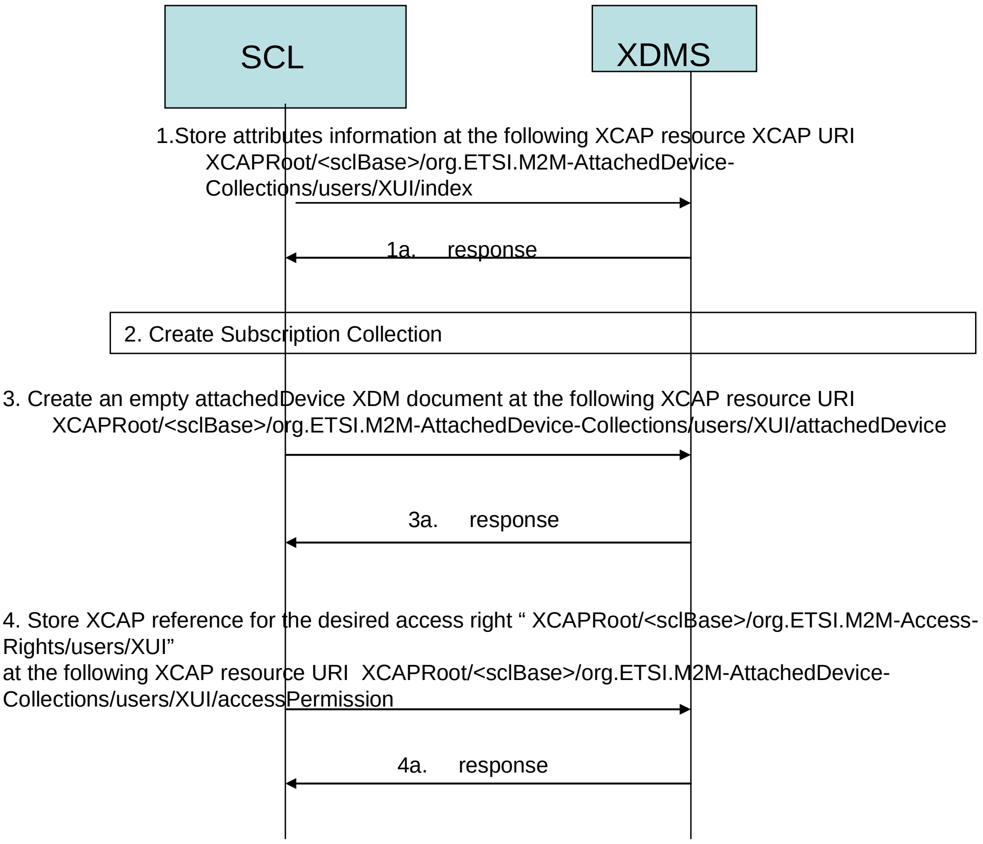

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty attachedDevice XDM document at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/attachedDevice
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI" at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/accessPermission
4a. response

Figure F.78: Attached Device Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the Network domain (NSCL) stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, Network domain (NSCL) creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 3, Network domain (NSCL) creates an empty XML document and store it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/attachedDevice.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4, the Network domain (NSCL) stores the XCAP resource that identifies the accessRight resource that applies to the Attached Device Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-AttachedDevice-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.

F.6.1.5 Creation of ExecInstance Collection

Figure F.79 shows the procedure to be performed by the NSCL in the Network domain to create an ExecInstance collection XDM resource.

In this case, the Network domain (NSCL) creates a new tree representing the ExecInstance collection resource under the users' tree as per the ExecInstance Collection application usage. The Network domain (NSCL) creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/) for the ExecInstance collection applying procedure (clause F.1.3.1.1).

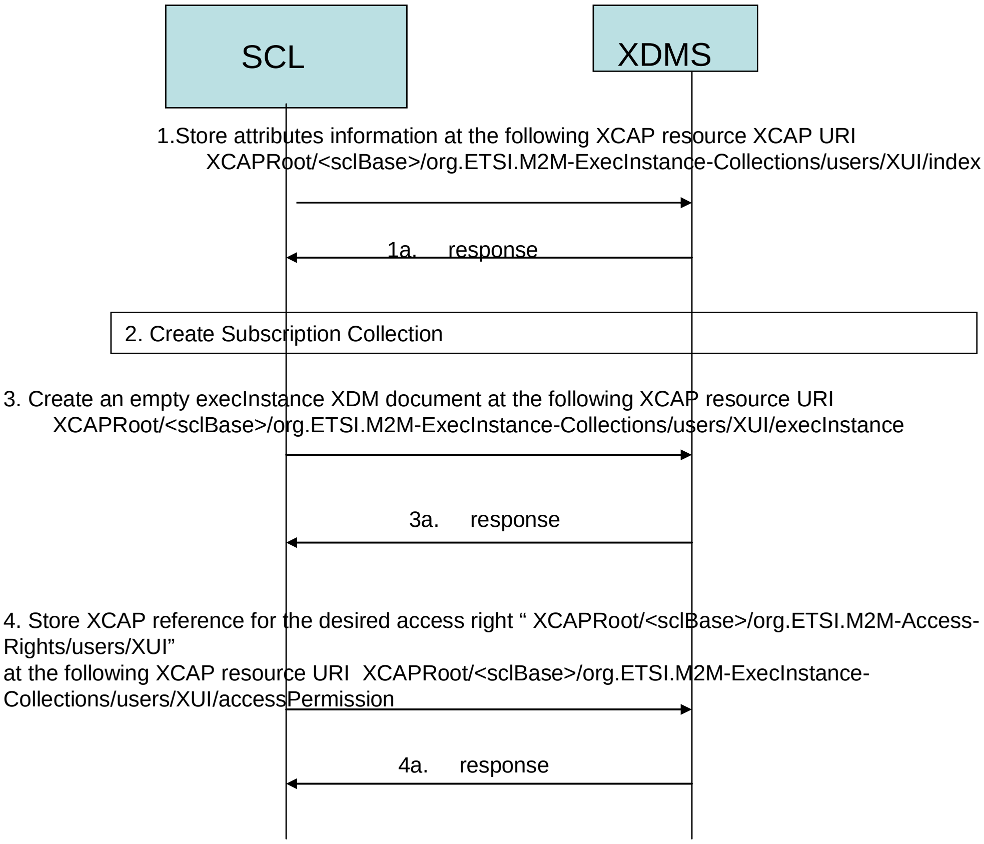

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty execInstance XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/execInstance
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/accessPermission
4a. response

Figure F.79: ExecInstance Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the Network domain (NSCL) stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, Network domain (NSCL) creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, Network domain (NSCL) creates an empty XML document and stores it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/execInstance.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the Network domain (NSCL) stores the XCAP resource that identifies the accessRight resource that applies to the Exec Instance Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-ExecInstance-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.

F.6.1.6 Creation of MgmtObj Collection

Figure F.80 shows the procedure to be performed by the NSCL in the Network domain to create a Management Object collection XDM resource.

In this case, the NSCL creates a new tree representing the Management Objects collection resource under the users' tree as per the MgmtObj Collection application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M- MgmtObj-Collections/users/XUI/) for the Management Object collection applying procedure (clause F.1.3.1.1).

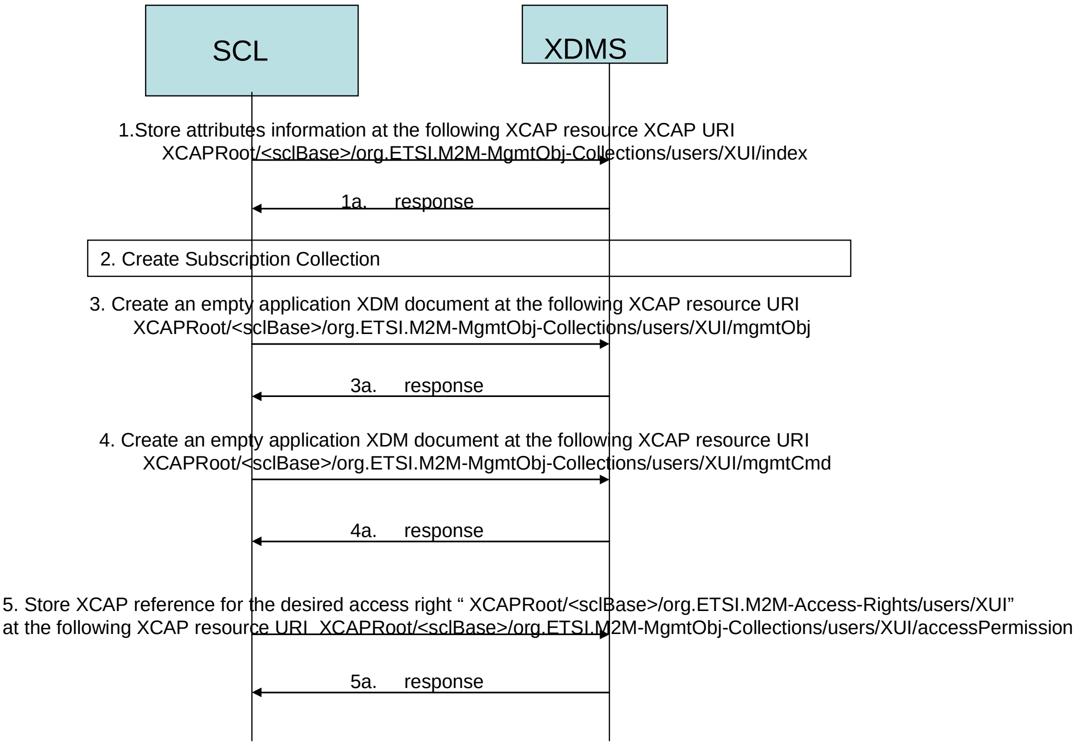

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty application XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/mgmtObj
3a. response
4. Create an empty application XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/mgmtCmd
4a. response
5. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/accessPermission
5a. response

Figure F.80: Management Object Collection Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL platform stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, Network domain (NSCL) creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, Network domain (NSCL) creates an empty XML document and store it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/mgmtObj.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, Network domain (NSCL) creates an empty XML document and store it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/mgmtCmd.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, the Network domain (NSCL) stores the XCAP resource that identifies the accessRight resource that applies to the Management Object Collection resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-MgmtObj-Collections/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.

- In step 5a, XDMS acknowledges the successful storage of the XDM document.

F.6.1.7 Creation of Parameters Resource

Figure F.81 shows the procedure to be performed by the NSCL in the Network domain to create a Parameters XDM resource.

In this case, the NSCL creates a new tree representing the Parameters resource under the users' tree as per Parameters application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M- Parameters/users/XUI/) for the Parameters applying procedure (clause F.1.3.1.1).

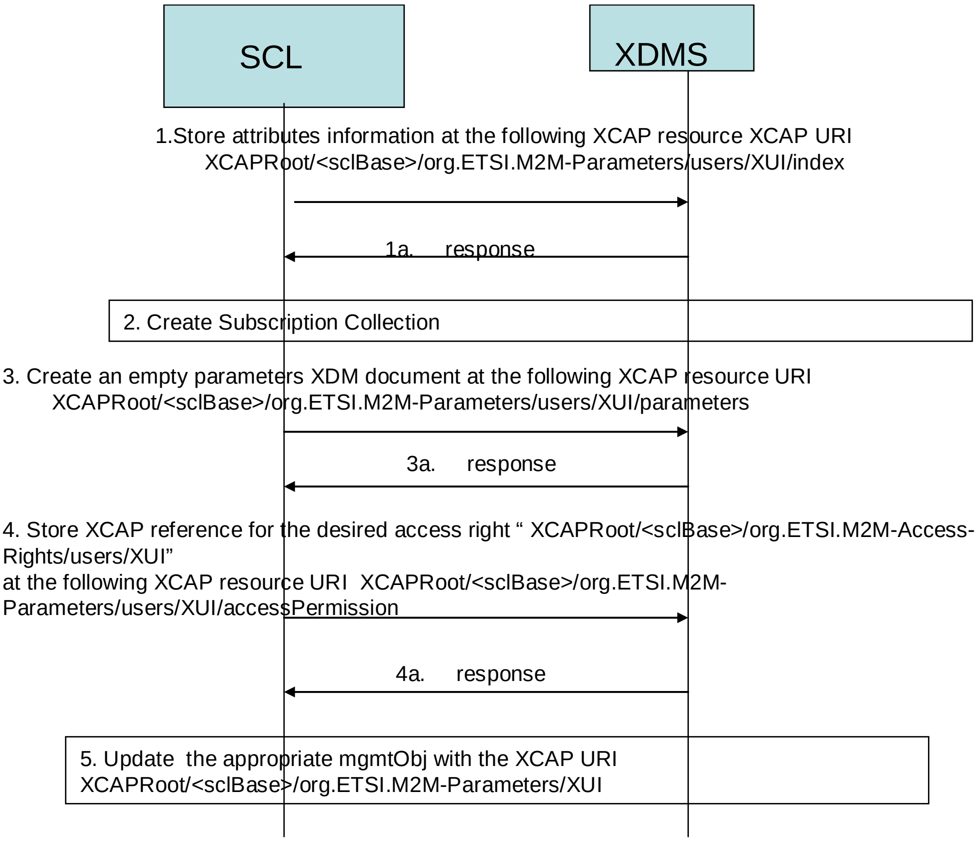

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty parameters XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/parameters
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/accessPermission
4a. response
5. Update the appropriate mgmtObj with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/XUI

Figure F.81: Parameter Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/index.

- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, Network domain (NSCL) creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 3, NSCL creates an empty XML document and store it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Paremeters/users/XUI/parameters.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4, NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Parameter resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Parameters/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate MgmtObj with the XCAP URI identifying the XUI.

F.6.1.8 Creation of Attached Device Resource

Figure F.82 shows the procedure to be performed by the NSCL in the Network domain to create an Attached Device XDM resource.

In this case, the NSCL creates a new tree representing the Attached Device resource under the users' tree as per Attached Device application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users/XUI/) for the attached device applying procedure (clause F.1.3.1.1).

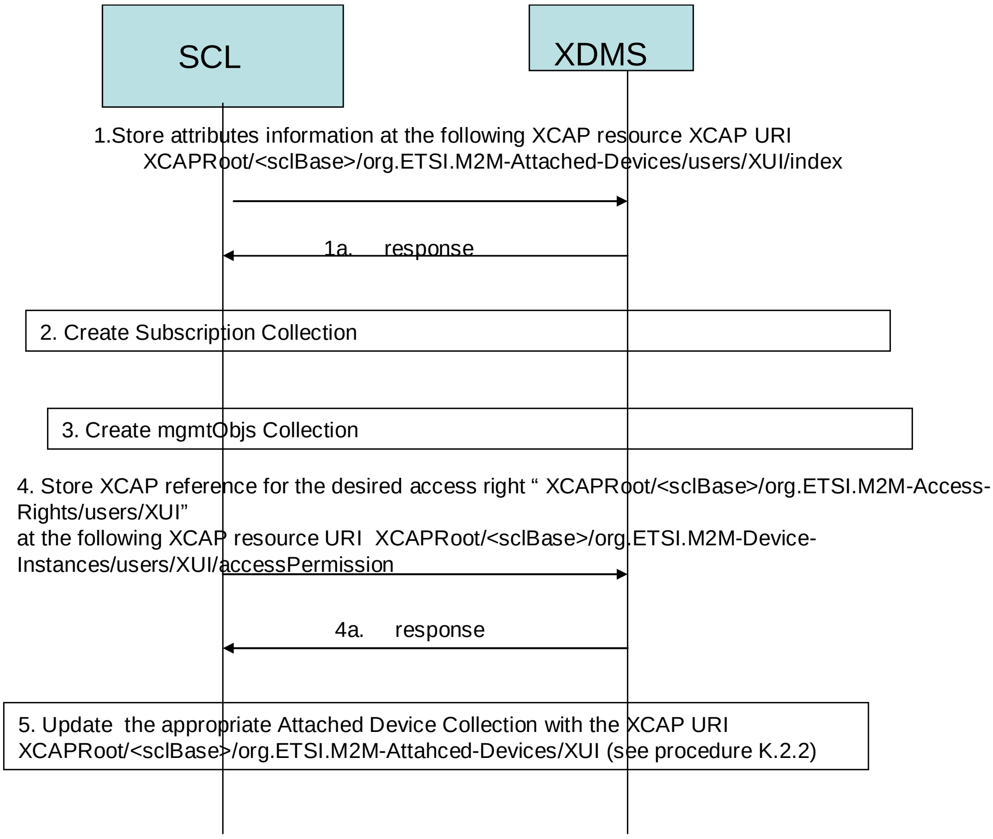

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create mgmtObjs Collection
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Device-Instances/users/XUI/accessPermission
4a. response
5. Update the appropriate Attached Device Collection with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Attahced-Devices/XUI (see procedure K.2.2)

Figure F.82: Attached Device Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL platform stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, NSCL creates a mgmtObjs collection resource using procedure (clause F.6.1.6) for that matter.
- In step 4, the Network domain (NSCL) stores the XCAP resource that identifies the accessRight resource that applies to the Attached Device resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Attached-Devices/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate attached Device Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.6.1.9 Creation of MgmtObj Resource

Figure F.83 shows the procedure to be performed by the NSCL in the Network domain to create a Management Object XDM resource.

In this case, the NSCL creates a new tree representing the Management Object resource under the users' tree as per Management Object application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/) for the Management Object applying procedure (clause F.1.3.1.1).

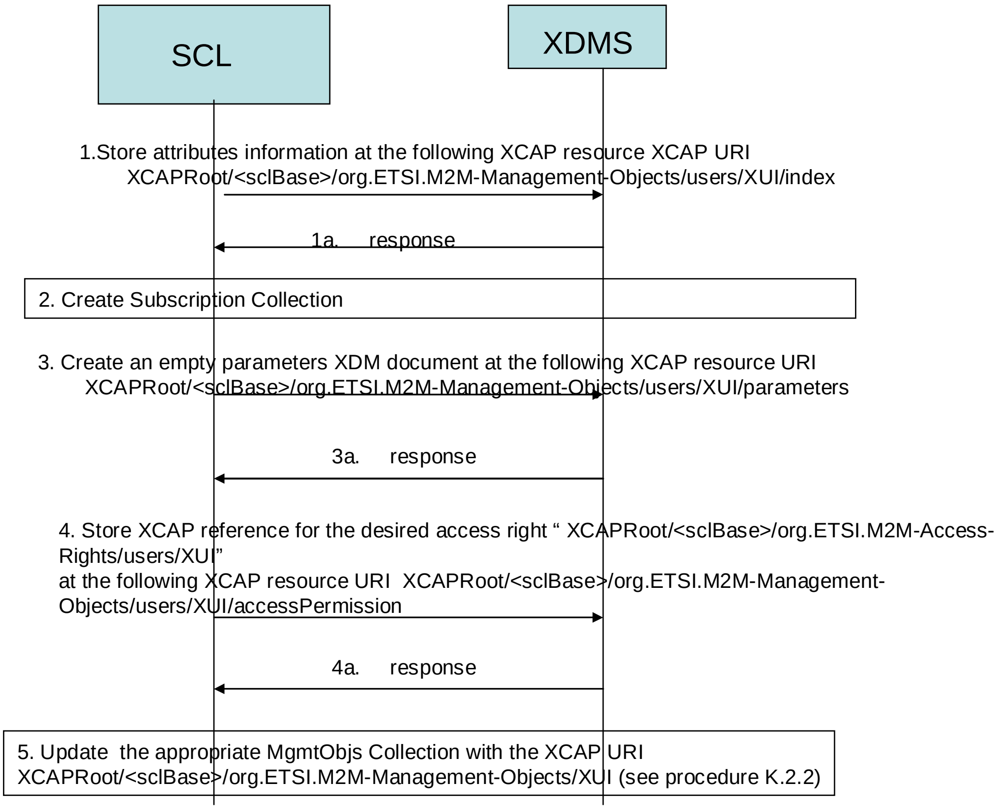

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create an empty parameters XDM document at the following XCAP resource URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/parameters
3a. response
4. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/accessPermission
4a. response
5. Update the appropriate MgmtObjs Collection with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/XUI (see procedure K.2.2)

Figure F.83: Management Object Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL platform stores all the attributes except the accessRight ID attribute in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, NSCL creates an empty XML document and store it at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/paremeters.
- In step 3a, XDMS acknowledges the successful storage of the XDM document.
- In step 4, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Management Object resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate MgmtObjs Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.6.1.10 Creation of MgmtCmd Resource

Figure F.84 shows the procedure to be performed by the NSCL in the Network domain to create a Management Command XDM resource.

In this case, the NSCL creates a new tree representing the Management Command resource under the users' tree as per Management Command application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/) for the Management Command applying procedure (clause F.1.3.1.1).

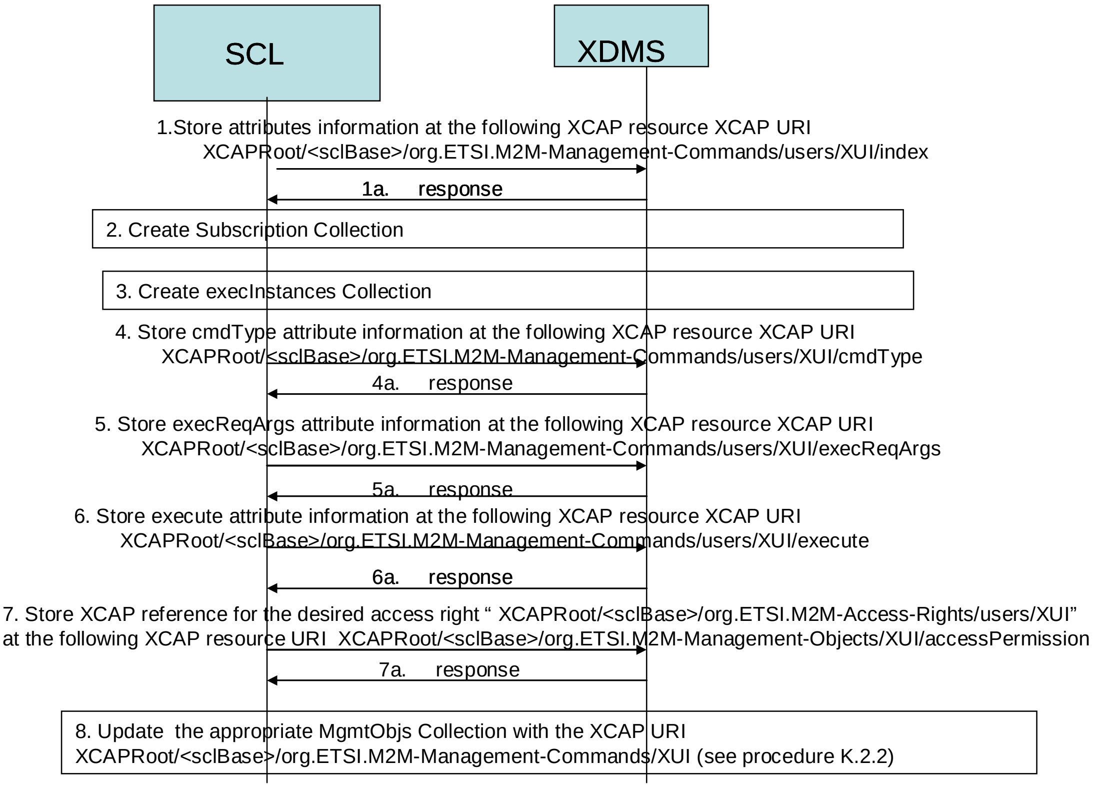

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/index
1a. response
2. Create Subscription Collection
3. Create execInstances Collection
4. Store cmdType attribute information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/cmdType
4a. response
5. Store execReqArgs attribute information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/execReqArgs
5a. response
6. Store execute attribute information at the following XCAP resource XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/execute
6a. response
7. Store XCAP reference for the desired access right " XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI"
at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/XUI/accessPermission
7a. response
8. Update the appropriate MgmtObjs Collection with the XCAP URI
XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/XUI (see procedure K.2.2)

Figure F.84: Management Commands Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL platform stores all the attributes except the accessRight ID attribute and other attributes defined below in subsequent steps in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.
- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.
- In step 3, NSCL creates an execInstances collection resource using procedure (clause F.6.1.5) for that matter.
- In step 4, NSCL stores cmdType attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/cmdType.
- In step 4a, XDMS acknowledges the successful storage of the XDM document.
- In step 5, NSCL stores execReqArgs attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/execReqArgs.

- In step 5a, XDMS acknowledges the successful storage of the XDM document.
- In step 6, NSCL stores execute attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/execute.
- In step 6a, XDMS acknowledges the successful storage of the XDM document.
- In step 7, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Management command resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Commands/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.
- In step 7a, XDMS acknowledges the successful storage of the XDM document.
- The NSCL updates the appropriate MgmtObjs Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

F.6.1.11 Creation of ExecInstance Resource

Figure F.85 shows the procedure to be performed by the NSCL in the Network domain to create an ExecInstances XDM resource.

In this case, the NSCL creates a new tree representing the execInstances resource under the users' tree as per ExecInstances application usage. The NSCL creates the XCAP URI (XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/) for the Management Command applying procedure (clause F.1.3.1.1).

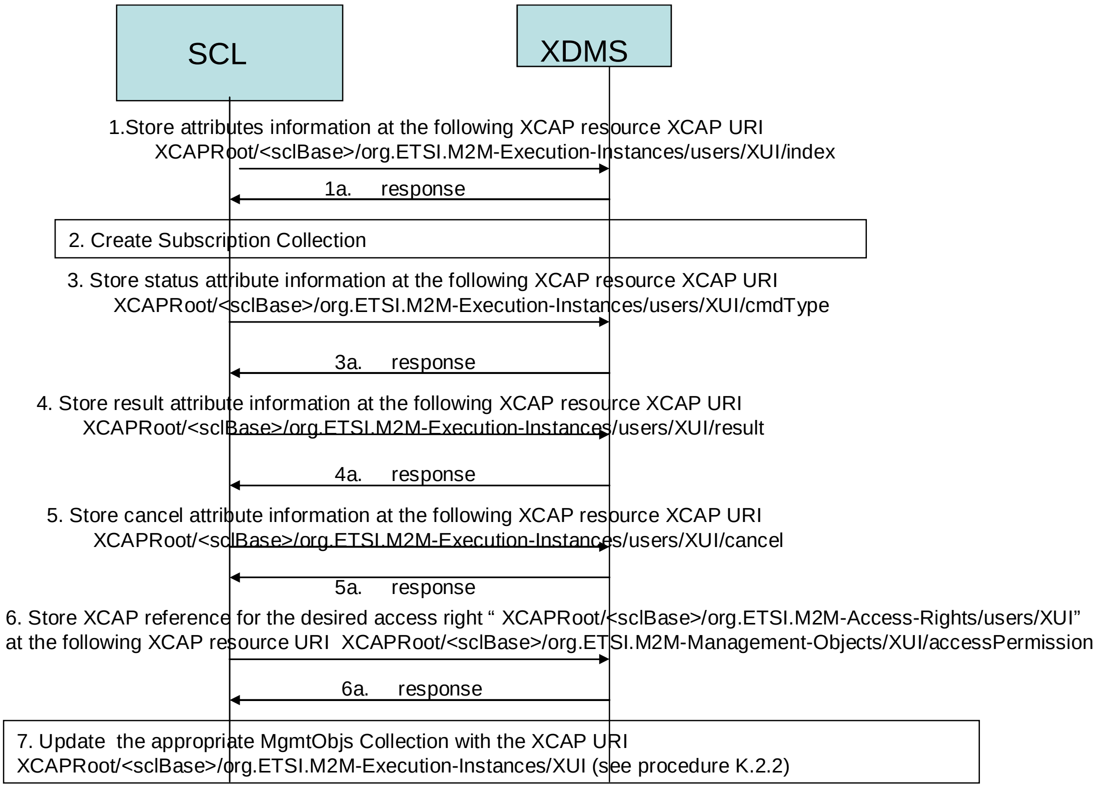

SCL
XDMS
1.Store attributes information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/index
1a. response
2. Create Subscription Collection
3. Store status attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/cmdType
3a. response
4. Store result attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/result
4a. response
5. Store cancel attribute information at the following XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/cancel
5a. response
6. Store XCAP reference for the desired access right “ XCAPRoot/<sclBase>/org.ETSI.M2M-Access-Rights/users/XUI” at the following XCAP resource URI XCAPRoot/<sclBase>/org.ETSI.M2M-Management-Objects/XUI/accessPermission
6a. response
7. Update the appropriate MgmtObjs Collection with the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/XUI (see procedure K.2.2)

Figure F.85: ExecInstances Resource creation process

The following is a brief description of the steps in the call flow:

- In step 1, the NSCL platform stores all the attributes except the accessRight ID attribute and attributes defined in subsequent steps in the index XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Executions-Instances/users/XUI/index.
- In step 1a, XDMS acknowledges the successful storage of the XDM document.

- In step 2, NSCL creates a subscription collection resource using procedure (clause F.2.1.19) for that matter.

- In step 3, NSCL stores status attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Commands/users/XUI/status.

- In step 3a, XDMS acknowledges the successful storage of the XDM document.

- In step 4, NSCL stores result attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Commands/users/XUI/result.

- In step 4a, XDMS acknowledges the successful storage of the XDM document.

- In step 5, NSCL stores cancel attribute information at the XCAP identified by the XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Commands/users/XUI/cancel.

- In step 5a, XDMS acknowledges the successful storage of the XDM document.

- In step 6, the NSCL stores the XCAP resource that identifies the accessRight resource that applies to the Execution Instance resource in the accessRight XDM document identified by the XCAP resource XCAP URI XCAPRoot/<sclBase>/org.ETSI.M2M-Execution-Instances/users/XUI/accessPermission. Note that the accessRight resource should have been created before that and the Network domain (NSCL) shall have the XCAP resource for it.

- In step 6a, XDMS acknowledges the successful storage of the XDM document.

- The NSCL updates the appropriate ExecInstance Collection with the XCAP URI identifying the XUI using procedure (clause F.2.2) for that matter.

Annex G (informative): Start-up and configuration operations

G.1 Start-up and configuration operations

The description of the following start-up and configuration operations reflects a specific way of implementing an SCL that can be in an ON and OFF state. This is only for the purpose of example and is not the only way to implement an SCL compliant system with the present document. Also there is no requirement that an SCL needs to have support for ON and OFF states.

In stage 3 the resources of an M2M entity are fully managed by the Service Capability Layer (SCL) in a storage here referenced as "*repository*", nevertheless even low computational power devices that have repository constraints (e.g. size of persistent memory) should be taken into account.

An SCL that follows the example implementation defined in this clause can be in one of the following operating states:

- **OFF:** when no operations are possible, e.g. after a power-off event.
- **ON:** when it is fully operative, e.g. after a power-on event.

When a new SCL in this example description is created (e.g. at first power-on event), that SCL should automatically perform the following local start-up and configuration operations that do not require actions on mId, mIa and dIa reference points:

- The resource tree is created in the repository starting from <sclBase> resource with all the required child resources.
- The content of the created resources is defined by a provided list of parameters.
- The <discovery> virtual resource is made available.

A previously created SCL subjected to a state transition from OFF to ON state, should automatically perform the following local start-up and configuration operations that do not require actions on mId, mIa and dIa reference points:

- Every resource should be checked for the expiration of the *expirationTime* attribute.
- Every content instance resource should be checked for the expiration of *maxInstanceAge* attribute defined in parent container resources.
- All expired resources should be deleted.

For low computational power devices that have persistent repository constraints (e.g. size of memory), the following cases may be identified:

- Transition from OFF to ON state: the resource tree is created in the same manner of a new SCL creation.
- Transition from ON to OFF state: it is suggested to save the SCL context into the available persistent memory in order to support the resource tree creation at the OFF to ON state transition.

Annex H (informative): Securing CoAP-based mId Using Object Security

When the mId interface is based on CoAP and an object security mechanism is desired, CoAP Security Options [i.2] is used. The present document identifies the keys and key names to be used with the XML and CoAP security. Other details are already governed by the [i.2] specifications.

Kmc and Kma are directly used by the CoAP security method [i.2].

Key Name for Kmc is set according to the following formula.

```
Key Name = "ETSI M2M Kmc " |
                <Value of Connection-ID in printable string form> | " " |
                <Value of Kmc index in printable string form>
```

For example: "ETSI M2M Kmc 1234567890 9999 33".

Annex I (informative):
Security Credential and Method Combinations

Figure I.1 provides an illustration of the various options for use as pre-provisioned credentials, M2M Bootstrap, M2M Connect, and mId security methods as described in the present document. Not all possible combinations are possible or meaningful among these options. All of the possible combinations can be generated from this figure by starting from a pre-provisioned credential type and following the arrows all the way down to the mId security methods.

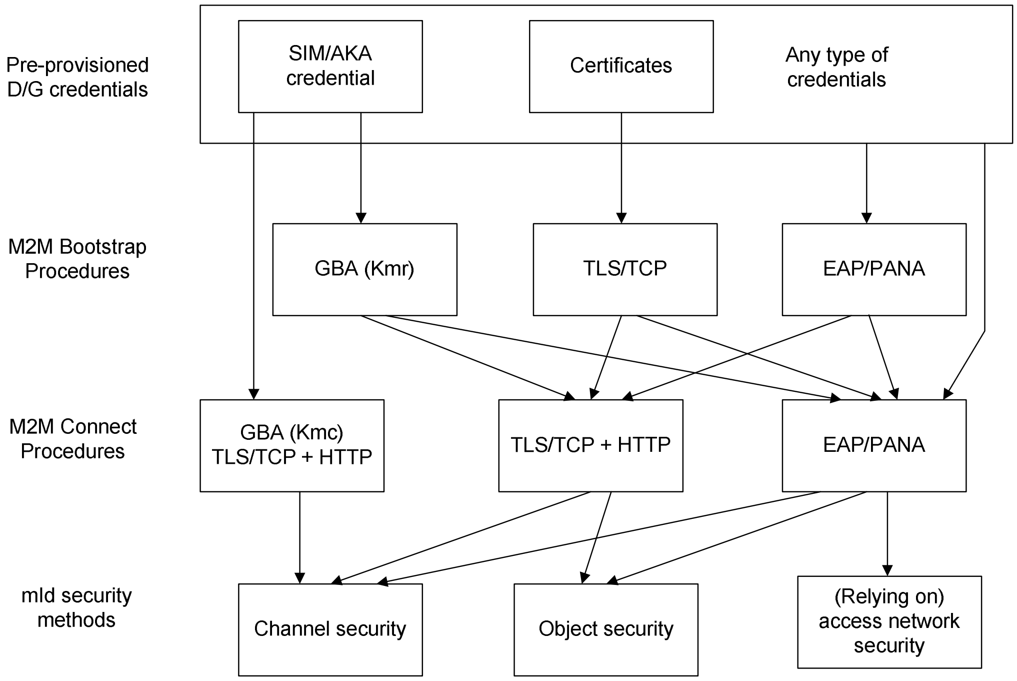


Figure I.1: Security credential and method combinations

Annex J (informative): Bibliography

IETF RFC 6655: "AES-CCM Cipher Suites for Transport Layer Security (TLS)".

History

Document history		
V1.1.1	February 2012	Publication
V1.2.1	June 2013	Publication