TTC仕様書
Technical Specification

TS-M2M-0004v3.11.2

oneM2M 技術仕様書
サービス層 API 仕様（共通部）

oneM2M Technical Specification
Service Layer Core Protocol

2019 年 06 月 28 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TS-M2M-0004v3.11.2

oneM2M 技術仕様書ーサービス層 API 仕様（共通部） [oneM2M Technical Specification - Service Layer Core Protocol]

＜参考＞ [Remarks]

１．英文記述の適用レベル [Application level of English description]

適用レベル [Application level]：E2

本標準の本文、付属資料および付録の文章および図に英文記述を含んでいる。

[English description is included in the text and figures of main body, annexes and appendices.]

２．国際勧告等の関連 [Relationship with international recommendations and standards]

本標準は、oneM2M で承認された Technical Specification 0004V3.11.2 に準拠している。

[This standard is standardized based on the Technical Specification 0004 (V3.11.2) approved by oneM2M.]

３．上記国際勧告等に対する追加項目等 [Departures from international recommendations]

原標準に対する変更項目 [Changes to original standard]

原標準が参照する標準のうち、TTC 標準に置き換える項目。

[Standards referred to in the original standard, which are replaced by TTC standards.]

原標準が参照する標準のうち、それらに準拠した TTC 標準等が制定されている場合は自動的に最新版 TTC 標準等に置き換え参照するものとする。

[Standards referred to in the original standard should be replaced by derived TTC standards.]

４．工業所有権 [IPR]

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページによる。

[Status of “Confirmation of IPR Licensing Condition” submitted is provided in the TTC web site.]

５．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

ONEM2M
TECHNICAL SPECIFICATION

Document Number	TS-0004-V3.11.2
Document Name:	Service Layer Core Protocol
Date:	2019-05-08
Abstract:	The present document specifies the communication protocol(s) for oneM2M compliant Systems, M2M Applications, and/or other M2M Systems. The present document also specifies common data formats, interfaces and message sequences to support reference points(s) defined by oneM2M.

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT PARTNER FOR THIS DOCUMENT, WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

Contents

List of figures

1 Scope

The present document specifies the communication protocol(s) for oneM2M compliant Systems, M2M Applications, and/or other M2M systems.

The present document also specifies the common data formats, interfaces and message sequences to support reference points(s) defined by oneM2M.

2 References

2.1 Normative 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 referenced document (including any amendments) applies.

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

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

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

[3] W3C Recommendation (2004): "XML Schema Part 2: Datatypes Second Edition".

[4] Void.

[5] Void.

[6] oneM2M TS-0001: "Functional Architecture".

[7] oneM2M TS-0003: "Security Solutions".

[8] IEEE 754-2008: "IEEE Standard for Floating-Point Arithmetic".

NOTE: Available at http://ieeexplore.ieee.org/servlet/opac?punumber=4610933.

[9] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings".

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

[11] IETF RFC 3987: "Internationalized Resource Identifiers (IRIs)".

[12] Void.

[13] Void.

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

[15] 3GPP TS 23.682: "Architecture enhancements to facilitate communications with packet data networks and applications".

[16] Void.

[17] 3GPP TS 23.003: "Numbering, addressing and identification".

[18] Void.

[19] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

[20] IETF RFC 5234: "Augmented BNF for Syntax Specifications: ABNF".

[21] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646".

[22] oneM2M TS-0008: "CoAP Protocol Binding".

[23] oneM2M TS-0009: "HTTP Protocol Binding".

[24] oneM2M TS-0010: "MQTT Protocol Binding".

[25] oneM2M TS-0011: "Common Terminology".

[26] IETF RFC 6838: "Media Type Specifications and Registration Procedures".

[27] ISO 8601:2004: "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[28] OMA-TS-REST_NetAPI_TerminalLocation: "Open Mobile Alliance; RESTful Network API for Terminal Location", Version 1.0.

[29] IETF RFC 4632: "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan".

[30] IETF RFC 5952: "A Recommendation for IPv6 Address Text Representation".

[31] 3GPP TS 32.299: "Telecommunication management; Charging management; Diameter charging applications".

[32] IETF RFC 4006: "Diameter Credit-Control Application".

[33] W3C Recommendation: "SPARQL 1.1 Query Language".

[34] W3C Recommendation: "RDF 1.1 XML Syntax".

[35] IETF RFC 4122: "A Universally Unique IDentifier (UUID) URN Namespace".

[36] oneM2M TS-0012: "oneM2M Base Ontology".

[37] oneM2M TS-0021: "oneM2M and AllJoyn Interworking".

[38] oneM2M TS-0022: "Field Device Configuration".

[39] IETF RFC 7049 (October 2013): "Concise Binary Object Representation (CBOR)".

[40] oneM2M TS-0023: "Home Appliances Information Model and Mapping".

[41] ISO 3166-1:2013: "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes".

[42] oneM2M TS-0020: "WebSocket Protocol Binding".

[43] oneM2M TS-0026: "3GPP Interworking".

[44] W3C Recommendation: "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)".

[45] W3C Recommendation: "OWL 2 Web Ontology Language XML Serialization (Second Edition)".

[46] W3C Recommendation: "OWL 2 Web Ontology Language: Mapping to RDF Graphs (Second Edition)".

[47] W3C Recommendation: "RDF 1.1 Turtle: Terse RDF Triple Language".

[48] W3C Note: "OWL 2 Web Ontology Language Manchester Syntax (Second Edition)".

[49] W3C Recommendation: "JSON-LD 1.1: A JSON-based Serialization for Linked Data".

[50] oneM2M TS-0034: "Semantics Support".

[51] 3GPP TS 29.122: "T8 reference point for Northbound Application Programming Interfaces (APIs)".

[49] IETF RFC 4566: "SDP: Session Description Protocol".

2.2 Informative 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 referenced document (including any amendments) applies.

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] oneM2M Drafting Rules.

NOTE: Available at http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf.

[i.2] Fielding, Roy Thomas (2000): "Architectural Styles and the Design of Network-based Software Architectures", Doctoral dissertation, University of California, Irvine.

[i.3] OMA-TS-REST-NetAPI-NotificationChannel: "Open Mobile Alliance; RESTful Network API for Notification Channel", OMA-TS-REST-NetAPI-NotificationChannel-V1-0.

[i.4] OMA-TS-MLP: "Open Mobile Alliance; Mobile Location Protocol", OMA-TS-MLP-V3-4-20130226-C Version 3.4.

[i.5] W3C Resource Description Framework.

NOTE: Available at https://www.w3.org/RDF/.

[i.6] W3C Recommendation: "SPARQL Query Language for RDF".

NOTE: Available at https://www.w3.org/TR/rdf-sparql-query/.

[i.7] IETF RFC 7515 (2015): "JSON Web Signature (JWS)".

[i.8] IETF RFC 7516 (2015): "JSON Web Encryption (JWE)".

[i.9] IETF RFC 7518 (2015): "JSON Web Algorithms (JWA)".

[i.10] IETF RFC 5771 (2010): "IANA Guidelines for IPv4 Multicast Address Assignments".

[i.11] IETF RFC 4291 (2006): "IP Version 6 Addressing Architecture".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in oneM2M TS-0011 Common Terminology [25] and the following apply:

Complex Data Type: data type that has a child element

Enumeration Type: data type that defines a variable to be one of a set of predefined constants

Group Hosting CSE: CSE where the addressed group resource resides

Hosting CSE: CSE where the addressed resource is hosted

Location Server: server offering location capabilities

M2M Area Network: network providing connectivity between Application Service Nodes or Application Dedicated Nodes and Middle Nodes in the field domain

Mca: Reference Point for M2M Communication with AE

Mcc: Reference Point for M2M Communication with CSE

Mcc': Reference Point for M2M Communication with CSE of different M2M Service Provider

NULL: *null* value

NOTE: The representation of *null* for different serialization types like xml and json can be found in clauses 8.3 and 8.4 respectively.

Originator: the AE/CSE that sends a Request

NOTE: In case of a request that traverses multiple reference points, the Originator is the AE/CSE that sends the first request in the sequence.

Receiver: entity that receives the Request

Receiver CSE: any CSE that receives a request

Registrar CSE: CSE where an Application or another CSE has registered

Registree/Registrar CSE: CSE that registers with another CSE

Request: message sent from the Originator to the Receiver

Response: message replying to the Request, sent from the Receiver to the Originator

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in oneM2M TS-0011 Common Terminology [25] and the following apply:

3GPP2	3rd Generation Partnership Project 2
ACP	AccessControlPolicy
AD	Anno Domini
AE-ID	Application Entity Identifier
ARC	Architecture
ASN-CSE	Application Entity that is registered with the CSE at Application Service Node
BCP	Best Current Practices
CDT	Common Data Type
CIDR	Classless Inter-Domain Routing
CMDH	Communication Management and Delivery Handling
CoAP	Constrained Application Protocol
CRUD	Create Retrieve Update Delete
CRUD+N	Create Retrieve Update Delete Notification
CSE-ID	Common Service Entity Identifier
CUDN	Create Update Delete Notify
DAA	Device Action Answer
DAR	Device-Action-Request
DNA	Device Notification Answer
DNR	Device Notification Request
DTLS	Datagram Transport Layer Security
FFS	For Further Study
FQDN	Fully Qualified Domain Name
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ID	IDentifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

IN-AE	Application Entity that is registered with the CSE in the Infrastructure Node
IN-CSE	CSE which resides in the Infrastructure Node
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MA	Mandatory Announced
MIME	Multipurpose Internet Mail Extension
MN-CSE	Reference Point for M2M Communication with CSE of different M2M Service Provider
MR	Mashup Requestor
MQTT	Message Queue Telemetry Transport
MTC-IWF	Machine Type Communications - InterWorking Function
NP	Not Present
NSE	Network Service Entity
OA	Optional Announced
OMA-DM	Open Mobile Alliance Device Management
RD	Retrieve Delete
RDF	Resource Description Framework
RFC	Request For Comment
RH	Resource Host
RPC	Remote Procedure Call
RSC	Response Status Codes
RUD	Retrieve Update Delete
SCS	Services Capability Server
SMF	Semantic Mashup Function
SMI	Semantic Mashup Instance
SMJP	Semantic Mashup Job Profile
SP	Service Provider
SPARQL	SPARQL Protocol and RDF Query Language
SP-ID	Service Provider Identifier
TBD	To Be Determined
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User Equipment
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UTF	UCS Transformation Format
UUID	Universally Unique Identifier
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Conventions

The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in the present document are to be interpreted as described in the oneM2M Drafting Rules [i.1].

To improve readability:

- The information elements of oneM2M Request/Response messages will be referred to as parameters. Parameter names will be written in bold italic.
- The information elements of resources will be referred to as attributes and child resources. Attributes will be written in italics.
- Abbreviated short names for information elements (see clause 8) will be written in bold italic.

5 Protocol design principles and requirements

5.1 General introduction

Clause 5 contains the design principles and requirements for the oneM2M protocol.

5.2 Introduction

5.2.0 Overview

The oneM2M architecture is resource-based (oneM2M TS-0001 [6]). The functionality of the system is exposed by means of APIs over all reference points specified in oneM2M TS-0001 [6]. Operations upon resources hosted by a CSE are carried over an established channel that constitutes the communication on the reference points Mca and Mcc. All resource operations could be fulfilled with the considerations in terms of scalability, extensibility, fault tolerance and robustness, energy efficiency, and self-operation.

Offline Charging using the Mch reference point is described in [6] and specified in Annex A.

Each resource operation comprises a pair of primitives: Request and Response. The Request and Response primitives can be represented as XML documents or JSON texts. This process of representing a primitive as XML documents, JSON texts or CBOR data format is denoted serialization in the present document. Serialization translates primitives into a format that can be stored or exchanged between network entities. This is exploited when transmitting primitives over communication protocols such as HTTP, CoAP, MQTT or WebSocket.

In order to provide a well-defined interface for the reference points in oneM2M TS-0001 [6], the following aspects need to be provided:

- the collection of primitives carried over a specific reference point; and
- the definitions and procedures of resource types in relation to the underlying protocols and reference points involved.

The present document provides:

- protocol design principles and requirements;
- data type definitions and representations;
- primitive format and generic procedures;
- common operations of originators and receivers;
- resource type-specific procedures; and
- XML definitions and schema.

NOTE: The actual binding of the interface to a specific protocol is not part of the present document, but is specified in a separate Technical Specifications [22], [23], [24], [42].

In accordance with the oneM2M architecture, each reference point is applicable to a wide range of underlying network technologies and transport protocols. oneM2M defines a set of bindings for specific underlying network technologies and transport protocols, these bindings are not limiting the applicability of the interfaces when used in other underlying networks and transport protocols. However, the behaviour of the interface needs to be respected in accordance to the present document and oneM2M TS-0001 [6].

5.2.1 Interfaces to the underlying networks

The CSEs access the network service functions provided by the underlying networks such as 3GPP, 3GPP2 and WLAN via Mcn reference point. Protocol bindings to the Mcn reference point are described in clause 9. The following services are provided by the underlying networks:

- Device triggering (see oneM2M TS-0026 [43])
- Location request (see Annex G)
- Device Management (see clause 7.3.4)

5.3 API design guidelines

The following are the guidelines for designing APIs:

1) APIs shall follow the principle of RESTful architecture, as described in [i.2].
2) APIs shall indicate which features are supported and not supported over the reference points specified in oneM2M TS-0001 [6].
3) APIs shall define how to address resources and how to manipulate resources, in accordance with oneM2M TS-0001 [6]; the resource is identified by a Universal Resource Identifier (URI) [2].
4) APIs shall provide the format and syntax of the operation primitives for all resources defined in oneM2M TS-0001 [6]. In case that for a particular protocol binding an operation cannot be supported it has to be clearly stated in the specific protocol binding technical specification.
5) Each Resource has a representation (see [i.2]) that shall be transferred and manipulated with the verbs. These verbs are identified as operations in oneM2M TS-0001 [6]: CREATE. RETRIEVE, UPDATE, DELETE and NOTIFY.
6) All primitives as well as the way that those primitives are sent shall be defined. The functionality of the primitives shall be compliant to the resource type specific procedure as specified in oneM2M TS-0001 [6], clause 10.2.
7) Primitives shall include attributes in accordance with oneM2M TS-0001 [6] for a specific resource.
8) Primitives shall be self-descriptive and contain all the information needed for the receiver of the primitives to handle the primitives.
9) Primitives should be idempotent operations which means no matter how many times the primitive is sent, the result does not change, in accordance to [i.2].
10) Primitives shall be mapped on the transport layer protocols.

5.4 Primitives

5.4.1 Introduction

Primitives are common service layer messages exchanged over the Mca, Mcc and Mcc' reference points.

There are two use cases:

- communication between an Originator and a Receiver which are collocated on the same M2M Node (e.g. ASN or MN) in the Common Service layer;
- communication between an Originator and a remote Receiver via an Underlying Network.

In the first use case the primitives may be exchanged directly between the Originator and Receiver processes.

In case of using an IP-based Underlying Network as illustrated in Figure 5.4.1-1, the primitives are mapped to application layer communication protocols such as HTTP, CoAP, MQTT or WebSocket which use TCP or UDP on the transport layer. The specification of primitives, however, is independent of underlying communication protocols and allows introduction of bindings to other communication protocols.

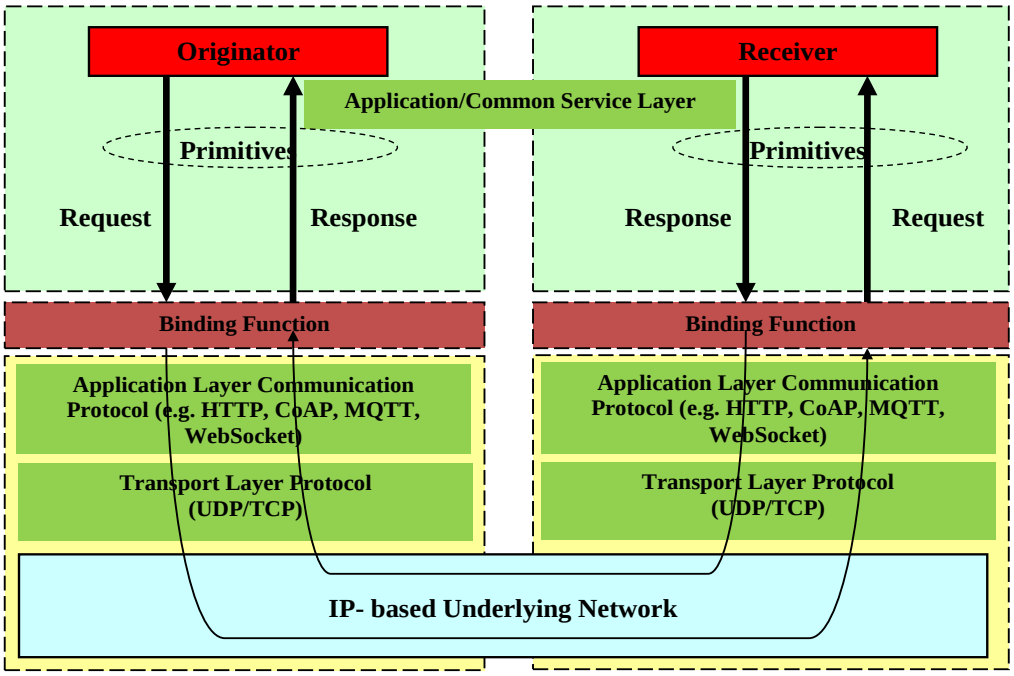

Figure 5.4.1-1: Communication model using Request and Response primitives over an IP-based Underlying Network

A single primitive in the common service layer may be mapped to zero or more transport messages by the communication protocol.

The Originators shall send requests to Receivers through primitives. The Originator and Receiver may be represented by either an AE or a CSE. The CRUD request primitive addresses a resource residing in a CSE. The Notify request primitive may address an AE or CSE.

Each CRUD+N operation consists of request and response primitives.

5.4.2 Primitives modelling

Primitives are modelled as follows.

A primitive is represented in the form of a data structure which defines, with appropriate parameters, specific procedures to be executed by both originator and receiver entities.

The data structure of a primitive consists of two parts:

- a control part, which contains parameters specifying the processing of the primitive; and
- an optional content part, which represents resource data, either the complete resource or only part of the resource (i.e. values of one or more resource attributes) in the partial addressing case.

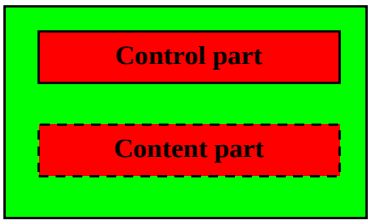

Figure 5.4.2-1: Primitive structure

5.4.3 Primitive principles

Execution of one primitive shall finish completely before execution of a subsequent primitive starts that affects the same resource.

When creating or updating the resource, its representation (full or partial) shall be contained in the content part of the primitive. Based on the representation of the resource, the Hosting CSE can create or update the entire resource without need for further information.

The operations on resources triggered by primitives shall be idempotent. This means no matter how many times the same primitive is targeted to the same resource, the resource does not change after the first execution of this primitive, with the exception of the creation of child resources.

5.4.4 Serialization of primitives

When transferred over a oneM2M reference point while using a communication protocol such as HTTP, CoAP, MQTT or WebSocket, the way oneM2M Request and Response primitives are represented shall be defined by a specific oneM2M protocol binding that is being used for the message transfer. The originator and receiver of each primitive use the same binding, and thus they will be using compatible serialization and deserialization techniques. Clause 8 of the present document defines canonical approaches for serializing primitives as JSON objects, XML documents or CBOR data format used by oneM2M protocol bindings.

5.5 Design principles

5.5.1 Introduction

The following clauses (5.5.2 to 5.5.7) present the design principles which wrap up the perspectives and ways in terms of definitions and procedures of APIs and resources for the oneM2M core protocol specified in the present document. These design principles shall cover all characteristics and advantages of oneM2M protocols including specifications of bindings to transport protocols such as HTTP, CoAP, MQTT and WebSocket.

The design of oneM2M protocols consider and mitigate the risk of unintended consequences, such as extensibility and interoperability issues, operational problems, or efficiency.

5.5.2 Extensibility

The oneM2M protocols are designed to allow continued development and to facilitate changes by means of standardized extensions.

The impact of the extensibility on the existing oneM2M protocol functions shall be minimized.

Extensibility can be related to one or more of the following aspects:

- handling a wide range of transport protocols as well as a different number of devices;
- adding, removing or modifying oneM2M protocol functionality;
- new oneM2M protocol routines;

- new primitives and data types.

5.5.3 Scalability

For provisioning scalability as a requirement in the design of oneM2M protocols, one or several of the following mechanisms are used:

- Ensuring direct addressability to the CSEs hosting target resources, to minimize network hops.
- Asynchrony in terms of data processing, with the objective of minimizing the number of discarded packets.
- Caching mechanisms that allow all the received packets to be processed.
- Efficient load distribution to avoid bottlenecks and data loss.
- Data compression and/or aggregation, in order to reduce the amount of data sent through the network.

5.5.4 Fault tolerance and robustness

One or more of the following mechanisms in terms of link availability can be exploited in the design of oneM2M protocols to account for a variety of exception conditions.

- To provide reliable transmission of data packets, packet recovery will be dealt with by using mechanisms appropriate for the operating environment (e.g. constrained devices, unreliable networks).
- When oneM2M protocols are employed over unreliable links, multiple data dissemination paths can be provided and maintained.

5.5.5 Efficiency

oneM2M protocols are designed with consideration of efficiency for networking involved resource-constrained devices.

- As energy consumption directly affects the overall system performance, oneM2M protocols should consider energy efficiency, especially in resource constrained environments with battery-powered oneM2M devices.
- Energy efficient oneM2M protocols aims at reducing the overall energy consumption while maintaining the performance required by the oneM2M Applications.

5.5.6 Inter-operability

API inter-operability between different protocol stacks is expected. For example, oneM2M API over HTTP/TCP/IP needs to inter-operate with CoAP/UDP in a local network using oneM2M API. oneM2M protocols are specified to provision the API inter-operability.

5.5.7 Self-operation and self-management

Devices employing the oneM2M API inter-work with established management protocols (e.g. security, discovery, bootstrapping, etc.). The inter-working with legacy management protocols via the oneM2M API shall be carried out in self-operation methods.

6 oneM2M protocols/API overview

6.1 Introduction

The present document describes message formats and procedures to communicate with a oneM2M-compliant M2M Platform System.

The present document describes:

- Data representation for communication protocol messages.
- Normal and exceptional procedure.
- Status codes.
- Guidelines for drafting APIs.

For wide acceptance by industrial markets, the present document describes structured and non-structured data for oneM2M Protocol using XML Schema Definition (XSD) language [3].

The actual format of data in request and response messages partially depends on the applied protocol binding. Mapping rules between the data formats defined in the present document types and protocol-specific native data formats are specified in the protocol binding specifications oneM2M TS-0008 [22], TS-0009 [23], TS-0010 [24] and TS-0020 [42].

The core data types of XML elements defined in the present document for use in oneM2M protocols shall use the namespace:

- http://www.onem2m.org/xml/protocols.

The present document, and any XML or XML Schema Documents produced by oneM2M shall use the prefix m2m: to refer to that namespace.

Specializations of the <flexContainer> resource type (see clause 7.4.37) may employ a different target namespace.

The XSD files referenced in the present document shall serve the following purposes:

11) Provide an unambiguous definition of XML element names and data types used for:
    - resource representations;
    - resource attributes;
    - Request and Response primitives (including Notification primitive);
    - parameters used in Request and Response primitives.
12) Help to identify and avoid multiple definition of equivalent data types with different names.
13) Provide a testable definition of the value range of data elements (e.g. allowed number range, allowed characters or character patterns, allowed enumeration values).

NOTE 1: The XML schemas do not fully check the value ranges of all data elements. This particularly applies to XML elements which represent string patterns (see Table 6.3.2-1). For full compliance with the present document, an implementation respects both the schema definition and any additional constraints given in the tables of data types defined in the present document.

14) Provide a testable definition of the presence of mandatory elements (minOccurs="1") and of cardinality of data elements (e.g. maxOccurs="2") in XML representations of data objects (i.e. resource instantiations and primitive parameters).

NOTE 2: The XSD files referenced in the present document are intended to validate instantiations of complete resources at the Hosting CSE. When requesting a CRUD operation and receiving the response, the ***Content*** primitive parameter however typically includes partial representations of a resource. Implementations compliant with the present document may employ modified versions of the XSD for schema-validation of partial resource representations.

15) Provide a testable definition of the correct sequence of occurrence of each element of a data object (where correct sequence is required).
16) Enable the use of development tools that generate executable code for data object processing from the XSD.
17) Enable the use of XML development tools which allow automatic generation of valid templates for XML and JSON objects, and validation of the compliance of any XML or JSON objects with the XSD.

Parameters and resource representations exchanged in primitives between oneM2M entities shall comply with data formats defined in the present document based on the referred XSD documents. The present document defines procedures for validation of received messages and the error handling in case of reception of non-compliant message content.

NOTE 3: M2M implementations are required to validate the data received in incoming primitives in accordance with the present document, but the present document does not intend to impose restrictions on implementation of the validation procedures. In particular the validation procedure is not required to use the XSD documents directly.

6.2 Addressing

6.2.1 Introduction

This clause describes the method of addressing oneM2M entities (e.g. AE or CSE) and oneM2M resources using identifiers described in the oneM2M TS-0001 [6].

6.2.2 Summary of oneM2M Identifiers

Table 6.2.2-1 shows the summary of M2M Identifiers defined in oneM2M TS-0001 [6].

Table 6.2.2-1: M2M Identifiers

Identifier	Data Type	Description
M2M-SP-ID	m2m:ID (see clause 6.3.3)	A globally unique ID as specified in [6]
App-ID	m2m:ID (see clause 6.3.3)	The identifier is specified in [6]
AE-ID	m2m:ID (see clause 6.3.3)	A globally unique ID as specified in [6]
CSE-ID	m2m:ID (see clause 6.3.3)	A globally unique ID as specified in [6]
M2M-Node-ID	m2m:nodeID (see clause 6.3.3)	A globally unique ID as specified in [6]
M2M-Sub-ID	m2m:ID (see clause 6.3.3)	A globally unique ID as specified in [6]
M2M-Request-ID	m2m:requestID (see clause 6.3.3)	A unique ID as specified in [6]
M2M-Ext-ID	m2m:externalID (see clause 6.3.3)	The identifier is specified in [6]
UNetwork-ID	m2m:ID (see clause 6.3.3)	A unique ID as specified in [6]
Trigger-Recipient-ID	m2m:triggerRecipientID	The identifier is specified in [6]
Role-ID	m2m:roleID (see clause 6.3.3)	A globally unique ID as specified in [6]
Token-ID	m2m:tokenID (see clause 6.3.3)	A globally unique ID as specified in [6]

6.2.3 oneM2M Entity Addressing

The oneM2M entities (e.g. AE or CSE) are identified and addressable using M2M Identifiers. Since an M2M Identifier is protocol independent, an IN-CSE shall accommodate address resolution functionality to get actual PoA addresses for communicating with other M2M entities using a specific protocol binding.

The present document assumes each oneM2M entity has the CSE-PoA address of its Registrar CSE in advance.

The CSE-ID shall be assigned by M2M Service Provider. The syntax of a CSE-ID is defined by the following ABNF notation [20]:

```
CSE-ID =  absolute-CSE-ID / SP-relative-CSE-ID
absolute-CSE-ID = M2M-SP-ID  SP-relative-CSE-ID
M2M-SP-ID = "//" FQDN
SP-relative-CSE-ID = "/" 1*unreserved
unreserved = (ALPHA / DIGIT) *(ALPHA / DIGIT / "-" / "." / "_")
FQDN = LABEL / (FQDN "." LABEL)
LABEL = LC-ALPHA [ *(LC_ALPHANUM / "-" ) LC-ALPHANUM ]
LC-ALPHANUM = %x61-7A / DIGIT
LC-ALPHA = %x61-7A
```

EXAMPLE 1 Starts:

EXAMPLE: `//myoperator.com/cse1`

This is an example of an absolute-CSE-ID

"//myoperator.com" is the M2M-SP-ID and "/cse1" is the SP-relative CSE-ID.

EXAMPLE 1 Ends.

The AE-ID is either assigned by the M2M Service Provider (S-type AE-ID Stem), or by the AE's Registrar CSE (C-Type Stem). The syntax of AE-ID in ABNF notation [20] is as follows:

```
AE-ID = absolute-AE-ID / SP-relative-AE-ID / S-AE-ID-Stem
absolute-AE-ID = M2M-SP-ID SP-relative-AE-ID
SP-relative-AE-ID = (SP-relative-CSE-ID "/" C-AE-ID-Stem ) / ("/" S-AE-ID-Stem )
S-AE-ID-Stem = "S" SP-assigned-AE-ID-Stem
C-AE-ID-Stem = "C" CSE-assigned-AE-ID-Stem
SP-assigned-AE-ID-Stem = 1*unreserved
CSE-assigned-AE-ID-Stem = 1*unreserved
```

EXAMPLE 2 Starts:

EXAMPLE: `//myoperator.com/S563423`

This is an example of an absolute-AE-ID that was assigned by the M2M-SP (//myoperator.com).

EXAMPLE: `//myoperator.com/cse2/C3532ea3`

This is an example of an absolute AE-ID, which registered on the Registrar CSE //myoperator.com/cse2. 'C3532ea3' is the AE-ID-Stem which is assigned by //myoperator.com/cse2.

EXAMPLE: `/cse2/C3532ea3`

This is the SP-relative version of the absolute AE-ID that is shown above.

EXAMPLE 2 Ends.

All M2M Identifiers are case-sensitive, so for example there could be two distinct CSEs one called `//myoperator.com/cse1` and the other one called `//myoperator.com/Cse1.`

Note that the M2M-SP-ID portion of an identifier contains the FQDN of the service provider. In general Domain Names are case-insensitive, but the FQDN component of an M2M Identifier shall always use lowercase characters as required by the ABNF and as shown in the examples given above.

6.2.4 oneM2M Resource Addressing

Authorized oneM2M entities can operate on a oneM2M resource by addressing the resource identifier as the target address (i.e. ***To*** parameter) in a request primitive. There are two resource addressing methods:

1) Structured resource identifier (used in Hierarchical Addressing): the identifier is constructed as a relative path from the <CSEBase> resource via parent resources.

2) Unstructured resource identifier (used in Non-hierarchical Addressing): the identifier uniquely identifies the resource in the domain of its Hosting CSE.

Virtual resource addressing is specified in clause 6.8.

Furthermore each resource identifier can be expressed in either

a. CSE-relative format, or

b. SP-relative format, or

c. Absolute format

A single attribute of the targeted oneM2M resource shall be addressable adding the sub-address, (`targeted-attribute-name`) following a "`#`" character after the resource address. This sub-address representing an attribute name shall be the short name (clause 8.2).

The syntax of the resource identifier in ABNF notion [20] is as follows:

```
resource-identifier = (structured-resource-identifier / unstructured-resource-identifier) [ "#"
targeted-attribute-name ]
structured-resource-identifier = [CSE-ID "/"] first-segment *("/" resource-name)
first-segment = resource-name / "-" / unstructured-CSE-relative-resource-identifier
unstructured-resource-identifier = [CSE-ID "/"] unstructured-CSE-relative-resource-identifier
unstructured-CSE-relative-resource-identifier = 1*unreserved
resource-name = 1*unreserved
```

If the resource-name is used in the first-segment production rule, it shall be the *resourceName* of the <CSEBase> resource. The character "-" (dash) can be used in the first-segment as a shortcut for the *resourceName* of the <CSEBase> resource.

When including resource identifiers into the Content parameter of response primitives (clause 7.5.2), the resource Hosting CSE shall use the CSE-relative format since the Originator knows the Hosting CSE ID.

All resource identifiers are case sensitive, so for example there could be two distinct resources one with identifier cin00856 and the other one with identifier CIN00856.

6.3 Common data types

6.3.1 Introduction

The following clauses (6.3.2 to 6.3.6) define the data format of resource attributes and parameters used in primitives.

All strings in oneM2M are case sensitive. Any primitive parameter or resource attribute name (or the name of any element contained therein) that has a datatype of xs:string or has a datatype derived from xs:string shall be treated as case-sensitive. In particular:

- An AE or CSE shall preserve the case of the characters in any strings that it processes.
- Any comparison of strings (for example the comparison made when evaluating a Filter Operation) shall take account of the case of the characters involved.

6.3.2 Simple data types incorporated from XML schema

The following "built-in data types" defined in Table 6.3.2-1 are incorporated from XML Schema definition [3].

Note that name space identifier for "http://www.w3.org/2001/XMLSchema" shall be referred to using the prefix xs: in the present document.

Table 6.3.2-1: Data Types incorporated from XML Schema

Data Type	Description	Notes
xs:anyType	A special complex type definition whose name is anyType in the XSD namespace, is present in each XSD schema. The definition of anyType serves as default type definition for element declarations whose XML representation does not specify one.	
xs:anySimpleType	The anySimpleType is considered to have an unconstrained lexical space for all built-in simple datatypes.	
xs:string	The string data type represents character strings in XML.	
xs:boolean	boolean represents the values of two-valued logic.	
xs:decimal	decimal represents a subset of the real numbers, which can be represented by decimal numerals. The value space of decimal is the set of numbers that can be obtained by dividing an integer by a non-negative power of ten, i.e. expressible as i / 10 n where i and n are integers and n ≥ 0. Precision is not reflected in this value space; the number 2.0 is not distinct from the number 2.00. The order relation on decimal is the order relation on real numbers, restricted to this subset.	
xs:float	The float data type is patterned after the IEEE single-precision 32-bit floating point data type IEEE 754-2008 [8]. Its value space is a subset of the rational numbers. Floating point numbers are often used to approximate arbitrary real numbers.	
xs:double	The double data type is patterned after the IEEE double-precision 64-bit floating point data type IEEE 754-2008 [8]. Each floating point data type has a value space that is a subset of the rational numbers. Floating point numbers are often used to approximate arbitrary real numbers.	
xs:duration	duration is a data type that represents durations of time.	
xs:hexBinary	hexBinary represents arbitrary hex-encoded binary data.	
xs:base64Binary	base64Binary represents arbitrary Base64-encoded binary data. For base64Binary data the entire binary stream is encoded using the Base64 Encoding defined in IETF RFC 4648 [9], which is derived from the encoding described in IETF RFC 2045 [10].	
xs:anyURI	anyURI represents an Internationalized Resource Identifier Reference (IRI). An anyURI value can be absolute or relative, and may have an optional fragment identifier (i.e. it may be an IRI Reference). This type should be used when the value fulfils the role of an IRI, as defined in IETF RFC 3987 [11] or its successor(s) in the IETF Standards Track.	
xs:normalizedString	normalizedString represents white space normalized strings. The value space of normalizedString is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters. The lexical space of normalizedString is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters. The base type of normalizedString is string.	
xs:token	token represents tokenized strings. The value space of token is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces. The lexical space of token is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces. The base type·of token is normalizedString.	
xs:NCName	The value space of NCName is the set of all strings which can be used as XML element names, omitting strings that contain ':' characters.	

Data Type	Description	Notes
xs:integer	integer is derived from decimal by fixing the value of fractionDigits to be 0 and disallowing the trailing decimal point. This results in the standard mathematical concept of the integer numbers. The value space of integer is the infinite set {...,-2,-1,0,1,2,...}. The base type of integer is decimal.	
xs:nonNegativeInteger	nonNegativeInteger has a lexical representation consisting of an optional sign followed by a non-empty finite-length sequence of decimal digits (#x30-#x39). If the sign is omitted, the positive sign ('+') is assumed. If the sign is present, it shall be "+" except for lexical forms denoting zero, which may be preceded by a positive ('+') or a negative ('-') sign. For example: 1, 0, 12678967543233, +100000.	
xs:positiveInteger	positiveInteger is derived from nonNegativeInteger by setting the value of minInclusive to be 1. This results in the standard mathematical concept of the positive integer numbers. The value space of positiveInteger is the infinite set {1,2,...}. The base type of positiveInteger is nonNegativeInteger.	
xs:unsignedLong	unsignedLong is derived from nonNegativeInteger by setting the value of maxInclusive to be 18446744073709551615. The base type of unsignedLong is nonNegativeInteger.	
xs:unsignedInt	unsignedInt is derived from unsignedLong by setting the value of maxInclusive to be 4294967295. The base type of unsignedInt is unsignedLong.	
xs:unsignedShort	unsignedShort is derived from unsignedInt by setting the value of maxInclusive to be 65535. The base type of unsignedShort is unsignedInt.	

6.3.3 oneM2M simple data types

Table 6.3.3-1 describes oneM2M-specific simple data type definitions. XML Schema data type definitions for these data types can be found in the XSD file called CDT-commonTypes-v3_11_0.xsd.

The types in Table 6.3.3-1 are either:

- Atomic data types derived from XML Schema data types by restrictions (other than enumeration) or union.
- List data types constructed from other XML Schema or oneM2M-defined atomic data types.

The oneM2M-defined enumeration data types are defined in clause 6.3.4.

Table 6.3.3-1: oneM2M Simple Data Types

XSD type name	Type Name	Examples	Description
m2m:resourceName	Resource name	myLightBulb 123Sensor	Used for resource name attribute. This shall be formed by (ALPHA / DIGIT) *(ALPHA / DIGIT / "-" / "." / "_") as described in clause 6.2.3
m2m:ID	Generic ID	//globalm2m.org	Used to represent generic IDs generated and used within oneM2M (M2M-SP-ID)
		//globalm2m.org/C190XX7T	(CSE-ID)
		//globalm2m.org/CSE1/123A38ZZY	(AE-ID)
m2m:nodeID	Node ID	urn:gsma:imei:90420156-025763-0;svn=42	Used for Node IDs. The constraints on this type are different from those on Generic IDs (IMEI as node ID)
m2m:deviceID	Device ID	urn:dev:ops:012345-Set%2DTop%2DBox-0123456789	A Device ID uniquely identifies a device using a URN. The format of the URN is one of IETF RFC 4122 [35] UUID, OPS URN, OS URN, IMEI URN, ESN URN, or MEID URN.A
m2m:externalID	M2M-EXT-ID	123456789@domain.com	The External Identifier allows the Underlying Network to identify the M2M Device (e.g. ASN, MN) associated with the CSE-ID or AE-ID. In the 3GPP case, the External Identifier is specified in 3GPP TS 23.003 [17]
	3GPP external Group ID	123456789@domain.com	In the 3GPP multicast case, the External Group Identifier is used in the group message delivery procedure and specified in 3GPP TS 23.682 [15]
m2m:requestID	Request ID	ab3f124a, CSE1/98821	Used for Request IDs. This type may include the ID of the target CSE as well as a part that varies for each ID
m2m:nhURI	Non Hierarchical Identifier	/CSE090112/ C190XX7T	Used where a resourceID is required to be non-hierarchical
m2m:acpType	List of ACP Resource IDs	//IN-CSEID.m2m.myoperator.org/93405	Used to represent a list of AccessControlPolicy identifiers. The list shall contain at least one member

XSD type name	Type Name	Examples	Description
m2m:labels	list of xs:token	printers networkwifi1 home_energy (key-only) domain:home color:red (key-value pair)	A list of tokens used for describing and discovering resources (searching wifi connected printer from vendor 1) Each token can have two formats, key-only and key-value pair. In the case of key-value pair, key and value are separated by ":". The key portion does not contain ":". The list shall contain at least one member
m2m:triggerRecipientID	Trigger Recipient Identifier	3010	Used when device triggering services are requested from the Underlying Network, to identify an instance of an ASN/MN-CSE on an execution environment, to which the trigger is routed. Defined as port number in the range 0 to 65535
m2m:listOfM2MID	List of M2M identifiers		xs:list of elements of data type m2m:ID. The list shall contain at least one member
m2m:listOfURIs	List of any URI		xs:list of elements of data type xs:anyURI. The list shall contain at least one member
m2m:listOfDuration	List of durations		xs:list of elements of data type xs:duration. The list shall contain at least one member
m2m:resourceTypeList	List of resource types		xs:list of elements of data type m2m:resourceType. The list shall contain at least one member
m2m:listOfMinMax	List of Time Limits	10 2560	xs:list of two xs:long values defining min and max limits of time intervals in units of milliseconds (value -1 representing infinite time)
m2m:ipv4	IPv4 address string with optional CIDR suffix	10.125.0.0/16,122.77.12.1	Used in m2m:accessControlRules specified in clause 6.3.5.27
m2m:ipv6	IPv6 address string with optional CIDR suffix	::/0, Fadf:ddd0::/32, abcd:ffff:abb0:aaaa::/64	Used in m2m:accessControlRules specified in clause 6.3.5.27
m2m:countryCode	Country Code	KR	2-character country code as defined by ISO 3166-1 [41]

XSD type name	Type Name	Examples	Description
m2m:pointOfAccess	single point of access of an AE or CSE	http://172.25.0.10:8080/xyz or coap://m2m.sp.com:5683 or mqtt://172.25.0.10:1883 or ws://10.222.254.26:80	A point of access is represented as a URI that shall contain the underlying transport protocol (in either lowercase or uppercase spelling), the IP address (or an FQDN in all lowercase) and optionally a port number and/or path. No whitespace characters are allowed. The protocol binding specifications may give additional instructions on how the URI is. interpreted
m2m:poaList	List of pointOfAccess strings	http://172.25.0.10:8080/xyz coap://m2m.sp.com:5683 mqtt://172.25.0.10:1883	xs:list of elements of data type m2m:pointOfAccess. The list shall contain at least one member
m2m:timestamp	Time stamp string	20141003T112032	DateTime string using 'Basic Format' specified in ISO 8601 [27]. Time zone shall be interpreted as UTC timezone. See below for more details
m2m:absRelTimestamp	absolute or relative time stamp string	20141003T112032 (absolute time),or 3600000 (relative time)	defined as xs:union of m2m:timestamp and xs:long data types
m2m:typeOfContent	Type of Content	application/xml	The media type shall be an IANA registered Media Types name, or an experimental Media Type (see [26]) ':'
m2m:serializations	Serialization types	application/xml application/json application/cbor	A list of IANA registered media types that can be used for serialization of primitives. The permitted values are: • application/xml • application/json • application/cbor The list shall contain at least one member
m2m:contentInfo	Content Information	application/xml:1 application/xml:1:0 application/xml:1:5	A string consisting of a media type followed by a m2m:encodingType and optional m2m:contentSecurity, each separated by ':' character. If the m2m:contentSecurity value is not present, then the preceding ':' shall also be not present. If the m2m:contentSecurity value is not present then this has the same interpretation as a value of 0 for m2m:contentSecurity. See note
m2m:protocolList	List of protocols	application/x-alljoyn;version=1.0 application/x-echonet-lite;version=1.0	A list of MIME types for all communication protocols supported by the device

XSD type name	Type Name	Examples	Description
m2m:eventCat	Event Category	2	Either: 1) one of the values from m2m:stdEventCats, or 2) a user-defined category in the range 100-999
m2m:eventCatWithDef	Event Category with default	0	Either: 1) a value from m2m:eventCat, or 2) the value 0 which has the special meaning "default"
m2m:listOfEventCat	List of (applicable) Event Categories	1 101	xs:list of elements of data type m2m:eventCat. The list shall contain at least one member
m2m:listOfEventCatWithDef	List of m2m:eventCatWithDef	0 1 101	xs:list of elements of data type m2m:eventCatWithDef. The list shall contain at least one member
m2m:scheduleEntry	Schedule Entry	* 0-5 2,6,10 * * * *	The string is used to describe a duration of enablement. The string format is described in clause 7.4.9.1
m2m:attributeList	List of xs:NCName	poa rr	Used for the ***Content*** parameter of Retrieve request primitives and in m2m:eventNotificationCriteria. Attributes represented with their short names. The list shall contain at least one member
m2m:roleID	Role-ID	1234abcd@role-issuer.com	A string pattern consisting of a name (the issuerRelativeID) and an FQDN in all lowercase (the issuerID) separated by the '@' character, not including any whitespace characters. The issuerRelativeID shall be comprised of any combination of the Roman alphabet, numerals, '.', '_' and '-' characters
m2m:sparql	SPARQL content	SELECT ?functionality WHERE { ?functionality rdf:type base:Measuring. ?functionality base:refersTo ?aspect. ?aspect rdf:type instance:Temperature }	The string is used for SPARQL content, e.g. in ***semanticsFilter***

XSD type name	Type Name	Examples	Description
m2m:missingDataList	List of absolute timestamp or list of relative timestamp	absolute time: 20141103T111832 20141103T112435 20141103T113633 or relative time: 10000 10005 10020	Used for storing the time information of missing data points in Time Series defined as xs:union of list of m2m:timestamp and list of xs:duration data types. The list shall contain at least one member
m2m:tokenID	Token-ID	1234abcd@token-issuer.com	A string pattern consisting of a name (the issuerRelativeID) and an FQDN in all lowercase (the issuerID) separated by the '@' character, not including any whitespace characters. See constraints above for the issuerRelativeID
m2m:dynAuthJWT	JSON Web Token (JWT), which uses either JSON Web Encryption (JWE) Compact Serialization JSON Web Signature (JWS) Compact Serialization	See m2m:e2eCompactJWE and m2m:e2eCompactJWS	Defined as xs:union of m2m:e2eCompactJWE and m2m:e2eCompactJWS
m2m:e2eCompactJWS	JSON Web Signature (JWS) Compact Serialization, used in End-to-End Security Features oneM2M TS-0003 [7]	eyJ0eXAiOiJK. eyJpc3MiOiJqb2UiLA0KIC. dBjftJeZ4CVP (line breaks for display purposes only)	Of the form [a].[b].[c], where components [a] and [c] are non-empty, while component [b] can be either empty or not empty. When not empty, each component is base64url encoded (IETF RFC 4648 [9]). See IETF RFC 7515 [i.7]
m2m:e2eCompactJWE	JSON Web Encryption (JWE) Compact Serialization, used in End-to-End Security Features oneM2M TS-0003 [7]	eyJ0eXAiOiJK. eyJpc3MiOiJqb2UiLA0KIC. dBjftJeZ4CVP. 5eym8TW_c8SuK. SdiwkIr3a. XFBoMYUZo (line breaks for display purposes only)	Of the form [a].[b].[c].[d].[e], where components [a] and [d] are non-empty, while components [b], [c] and [e] can be empty or not empty. When not empty, each component is base64url encoded (IETF RFC 4648 [9]). See IETF RFC 7516 [i.8]
m2m:signatureList	List of signatures generated using a certificate or MIC generated using a symmetric key. It is used in Authorization Relationship Mapping	i6watmQQQ1y3GB- VsWq5fJKzQcBB4jRfH1bfJFj0JtFVtLotttzY yA== IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWo T4t0_gyTE96639In0FZFY2_rvP- _bMJ01EArmKZsR5VW3rwoPxw== (line breaks for display purposes only)	Each signature or MIC in the list is represented as a string which is base64url encoded(IETF RFC 4648 [9]). The list shall contain at least one member

XSD type name	Type Name	Examples	Description
m2m:locationTargetID	The identifier to be used for retrieving the location information of a remote Node or device of underlying network	urn:gsma:imei:90420156-025763-0;vers=0 or 123456789@domain.com;svn=42 or 8617791450839	defined as xs:union of m2m:nodeID and m2m:externalID and MSISDN
m2m:releaseVersion	Service Layer Release Version	3 or 2a	This parameter is set to the release version that the primitive complies with
m2m:supportedReleaseVersions	List of supported Release Versions	applicable list elements: 1, 2, 2a, 3	This list includes the release versions supported by AE or CSE. The list shall contain at least one member
m2m:TMGI	Temporary Mobile Group Identity allocated to the MBMS bearer.	F2003090156	A string assigned by the 3GPP network used to identify the MBMS Bearer Service. The format is defined in 3GPP TS 23.003 [17]
m2m:sessionDescription	Session Description	o=user 2890844526 2890844526 IN IP4 10.1.1.1 s=stream c=IN IP4 10.1.1.1 t=0 0 m=video 5600 RTP/AVP 96 a=rtpmap:96 H264/90000	The description format is a multi-lined text string as defined in Session Description Protocol (IETF RFC 4566 [52])
m2m:sessionCapabilities	Session Capability	audio:AVP video:RTS/AVP	Pair(s) of media type and corresponding protocol as defined in Session Description Protocol (IETF RFC 4566 [52]). The delimiter between the media type and the protocol is colon
NOTE: The media type and m2m:encodingType in m2m:contentInfo describe the content data to which the End-to-End Security of Data (ESData) processing, if any, was applied as indicated by m2m:contentSecurity. The m2m:contentInfo indicates a sequence of processes to be applied to the *content* after being obtained from the CSE. First, the ESData processing (if any) as indicated by m2m:contentSecurity is applied. The result of this processing then has transfer decoding (if any) applied as indicated by m2m:encodingType. The result of this processing is then processed according to the media type.			

The m2m:timestamp datatype uses ISO 8601 [27] Complete Representation using the Basic Format as described here:

- The timestamp shall be a string containing Year, Month, Day, Hours, Minutes and Seconds components using the format YYYYMMDDThhmmss as defined in [27]. In this representation the character "T" is to indicate the start of the time of day portion.
- All these components shall appear in the string; reduced representations are not permitted.
- The Seconds component may optionally contain a decimal fraction. In this case the string shall contain two integer digits, followed by a comma and then one or more fractional digits, up to a maximum of six. For example YYYYMMDDThhmmss,ssssss
- The timestamp string shall not contain Timezone information. All timestamps shall be interpreted as being in UTC.

A receiving or Hosting CSE shall accept a timestamp that contains fractional seconds, but it need not act on a timestamp with the level of precision that is implied by its fractional part. For example, it is acceptable for a Hosting CSE to round up an expiration time when interpreting it.

NOTE 1: Care should be taken when developing an application that compares timestamps. This is because AEs and CSEs are not required to have their clocks synchronized.

NOTE 2: As the m2m:timestamp is expressed in UTC, an AE has to be aware of the Timezone in which it is operating if it is to be able to relate the timestamp to its local time.

6.3.4 oneM2M enumerated data types

6.3.4.1 Introduction

The oneM2M Enumeration Types are defined as extension from "enumeration type" which is defined in XML Schema definition [3]. The oneM2M Enumeration Types are based on xs:integer, and the numeric values are interpreted as specified in clause 6.3.4.2. Table 6.3.4.1-1 shows the example Enumeration Type definition for m2m:enumFooType.

Table 6.3.4.1-1: Example of oneM2M Enumeration Type Definition

Value	Interpretation	Note
1	Interpretation-1	
2	Interpretation-2	
3	Interpretation-3	
NOTE: See clause x.x.x "title of clause".		

The oneM2M Enumeration Type definition shall be implemented as part of CDT-enumeration-v3_11_0.xsd. Figure 6.3.4.1-1 shows the example XSD representation of "m2m:enumFooType".

```
<xs:simpleType name="enumFooType">
 <xs:restriction base="xs:integer">
  <xs:enumeration value="1"/>
  <xs:enumeration value="2"/>
  <xs:enumeration value="3"/>
 </xs:restriction>
</xs:simpleType>
```

Figure 6.3.4.1-1: Example of XSD version of oneM2M Enumeration Type

6.3.4.2 Enumeration type definitions

6.3.4.2.1 m2m:resourceType

Table 6.3.4.2.1-1: Interpretation of resourceType

Value	Interpretation	Note
1	accessControlPolicy	
2	AE	
3	container	
4	contentInstance	
5	CSEBase	
6	delivery	
7	eventConfig	
8	execInstance	
9	group	
10	locationPolicy	
11	m2mServiceSubscriptionProfile	
12	mgmtCmd	
13	mgmtObj	
14	node	
15	pollingChannel	
16	remoteCSE	
17	request	
18	schedule	

Value	Interpretation	Note
19	serviceSubscribedAppRule	
20	serviceSubscribedNode	
21	statsCollect	
22	statsConfig	
23	subscription	
24	semanticDescriptor	
25	notificationTargetMgmtPolicyRef	
26	notificationTargetPolicy	
27	policyDeletionRules	
28	flexContainer	
29	timeSeries	
30	timeSeriesInstance	
31	role	
32	token	
33	void	
34	dynamicAuthorizationConsultation	
35	authorizationDecision	
36	authorizationPolicy	
37	authorizationInformation	
38	ontologyRepository	
39	ontology	
40	semanticMashupJobProfile	
41	semanticMashupInstance	
42	semanticMashupResult	
43	AEContactList	
44	AEContactListPerCSE	
45	localMulticastGroup	
46	multimediaSession	
47	triggerRequest	
48	crossResourceSubscription	
49	backgroundDataTransfer	
50	transactionMgmt	
51	transaction	
10001	accessControlPolicyAnnc	
10002	AEAnnc	
10003	containerAnnc	
10004	contentInstanceAnnc	
10009	groupAnnc	
10010	locationPolicyAnnc	
10013	mgmtObjAnnc	
10014	nodeAnnc	
10016	remoteCSEAnnc	
10018	scheduleAnnc	
10024	semanticDescriptorAnnc	
10028	flexContainerAnnc	
10029	timeSeriesAnnc	
10030	timeSeriesInstanceAnnc	
10033	void	
10034	dynamicAuthorizationConsultationAnnc	
10038	ontologyRepositoryAnnc	
10039	ontologyAnnc	
10040	semanticMashupJobProfileAnnc	
10041	semanticMashupInstanceAnnc	
10042	semanticMashupResultAnnc	
10046	multimediaSessionAnnc	
NOTE: See clause 6.4.1 Request primitive parameter data types.		

6.3.4.2.2 m2m:cseTypeID

Used for the *cseType* attribute of the <CSEBase> resource.

Table 6.3.4.2.2-1: Interpretation of cseTypeID

Value	Interpretation	Note
1	IN_CSE	
2	MN_CSE	
3	ASN_CSE	
NOTE: See clause 7.4.4 "Resource Type remoteCSE".		

6.3.4.2.3 m2m:locationSource

Used for the *locationSource* attribute of the <locationPolicy> resource.

Table 6.3.4.2.3-1: Interpretation of locationSource

Value	Interpretation	Note
1	Network_based	
2	Device_based	
3	Sharing_based	
NOTE: See clause 7.4.10 "Resource Type locationPolicy".		

6.3.4.2.4 m2m:stdEventCats

Used for the ***Event Category*** parameter in the request primitive and the *eventCat* attribute of the <delivery> resource and the cmdh policy resource types.

Table 6.3.4.2.4-1: Interpretation of stdEventCats

Value	Interpretation	Note
2	Immediate	
3	BestEffort	
4	Latest	
NOTE: See clause 7.4.11 "Resource Type delivery" and clause D.12 "Resource cmdhPolicy".		

6.3.4.2.5 m2m:operation

Used for the ***Operation*** parameter in request and the *operation* attribute of the <request> resource.

Table 6.3.4.2.5-1: Interpretation of operation

Value	Interpretation	Note
1	Create	
2	Retrieve	
3	Update	
4	Delete	
5	Notify	
NOTE: See clause 6.4.1 Request primitive parameter data types.		

6.3.4.2.6 m2m:responseType

Used for ***Response Type*** parameter (as a part of responseTypeInfo, see clause 6.3.5.30) in the request primitive.

Table 6.3.4.2.6-1: Interpretation of responseType

Value	Interpretation	Note
1	nonBlockingRequestSynch	
2	nonBlockingRequestAsynch	
3	blockingRequest	
4	flexBlocking	
5	noResponse	This shall only be used for procedures related to 3GPP Interworking defined in oneM2M TS-0026 [43].
NOTE: See clause 6.4.1 Request primitive parameter data types.		

6.3.4.2.7 m2m:resultContent

Used for ***Result Content*** parameter in the request primitive.

Table 6.3.4.2.7-1: Interpretation of resultContent

Value	Interpretation	Note
0	nothing	
1	attributes	
2	hierarchical address	
3	hierarchical address and attributes	
4	attributes and child resources	
5	attributes and child resource references	
6	child resource references	
7	original resource	
8	child resources	
9	modified attributes	
10	semantic content	
NOTE: See clause 6.4.1 Request primitive parameter data types.		

6.3.4.2.8 m2m:desIdResType

Used in the *metainformation* attribute of the <request> resource.

Table 6.3.4.2.8-1: Interpretation of desIdResType

Value	Interpretation	Note
1	structured	
2	unstructured	
NOTE: See clause 6.4.1 Request primitive parameter data types.		

6.3.4.2.9 m2m:responseStatusCode

The values for this data type are defined in clause 6.6.3 "Definition of Response Status Codes".

Table 6.3.4.2.9-1: Interpretation of responseStatusCode

Value	Interpretation	Note
("Numeric Code" in clause 6.6.3)	("Description" in clause 6.6.3)	

6.3.4.2.10 m2m:requestStatus

Used for the *requestStatus* attribute of the <request> resource.

Table 6.3.4.2.10-1: Interpretation of requestStatus

Value	Interpretation	Note
1	COMPLETED	
2	FAILED	
3	PENDING	
4	FORWARDED	
5	PARTIALLY_COMPLETED	
NOTE: See clause 7.4.12 "Resource Type request".		

6.3.4.2.11 m2m:memberType

Used for the *memberType* attribute of the <group> resource.

Table 6.3.4.2.11-1: Interpretation of memberType

Value	Interpretation	Note
0	mixed	A mixture of all the resource types (except mixed itself).
1	accessControlPolicy	
2	AE	
3	container	
4	contentInstance	
5	CSEBase	
6	delivery	
7	eventConfig	
8	execInstance	
9	group	
10	locationPolicy	
11	m2mServiceSubscription	
12	mgmtCmd	
13	mgmtObj	
14	node	
15	pollingChannel	
16	remoteCSE	
17	request	
18	schedule	
19	serviceSubscribedAppRule	
20	serviceSubscribedNode	
21	statsCollect	
22	statsConfig	
23	subscription	
24	semanticDescriptor	
25	notificationTargetMgmtPolicyRef	
26	notificationTargetPolicy	
27	policyDeletionRules	
28	flexContainer	
29	timeSeries	
30	timeSeriesInstance	
31	role	
32	token	
33	void	
34	dynamicAuthorizationConsultation	
35	authorizationDecision	
36	authorizationPolicy	
37	authorizationInformation	
38	ontologyRepository	

Value	Interpretation	Note
39	ontology	
40	semanticMashupJobProfile	
41	semanticMashupInstance	
42	semanticMashupResult	
43	AEContactList	
44	AEContactListPerCSE	
46	multimediaSession	
47	triggerRequest	
48	crossResourceSubscription	
49	backgroundDataTransfer	
50	transactionMgmt	
51	transaction	
10001	accessControlPolicyAnnc	
10002	AEAnnc	
10003	containerAnnc	
10004	contentInstanceAnnc	
10009	groupAnnc	
10010	locationPolicyAnnc	
10013	mgmtObjAnnc	
10014	nodeAnnc	
10016	remoteCSEAnnc	
10018	scheduleAnnc	
10024	semanticDescriptorAnnc	
10028	flexContainerAnnc	
10029	timeSeriesAnnc	
10030	timeSeriesInstanceAnnc	
10033	void	
10034	dynamicAuthorizationConsultationAnnc	
10038	ontologyRepositoryAnnc	
10039	ontologyAnnc	
10040	semanticMashupJobProfileAnnc	
10041	semanticMashupInstanceAnnc	
10042	semanticMashupResultAnnc	
10046	multimediaSessionAnnc	
20001	oldest	
20002	latest	
20003	mashup	
NOTE: See clause 7.4.13 "Resource Type group".		

6.3.4.2.12 m2m:consistencyStrategy

Used for the *consistencyStrategy* attribute of the <group> resource.

Table 6.3.4.2.12-1: Interpretation of consistencyStrategy

Value	Interpretation	Note
1	ABANDON_MEMBER	
2	ABANDON_GROUP	
3	SET_MIXED	
NOTE: See clause 7.4.13 "Resource Type group".		

6.3.4.2.13 m2m:cmdType

Used for the *cmdType* attribute of the <mgmtCmd> resource.

Table 6.3.4.2.13-1: Interpretation of cmdType

Value	Interpretation	Note
1	RESET	
2	REBOOT	
3	UPLOAD	
4	DOWNLOAD	
5	SOFTWAREINSTALL	
6	SOFTWAREUNINSTALL	
7	SOFTWAREUPDATE	
NOTE: See clause 7.4.16 "Resource Type mgmtCmd".		

6.3.4.2.14 m2m:execModeType

Used for the *execModeType* attribute of <mgmtCmd> and <execInstance> resources.

Table 6.3.4.2.14-1: Interpretation of execModeType

Value	Interpretation	Note
1	IMMEDIATEONCE	
2	IMMEDIATEREPEAT	
3	RANDOMONCE	
4	RANDOMREPEAT	
NOTE: See clause 7.4.16 "Resource Type mgmtCmd" and clause 7.4.17 "Resource Type execInstance".		

6.3.4.2.15 m2m:execStatusType

Used for the *execStatusType* attribute of the <execInstance> resource.

Table 6.3.4.2.15-1: Interpretation of execStatusType

Value	Interpretation	Note
1	INITIATED	
2	PENDING	
3	FINISHED	
4	CANCELLING	
5	CANCELLED	
6	STATUS_NON_CANCELLABLE	
NOTE: See clause 7.4.17 "Resource Type execInstance".		

6.3.4.2.16 m2m:execResultType

Used for the *execResult* attribute of the <execInstance> resource.

Table 6.3.4.2.16-1: Interpretation of execResultType

Value	Interpretation	Note
0	STATUS_SUCCESS	
1	STATUS_REQUEST_UNSUPPORTED	
2	STATUS_REQUEST DENIED	
3	STATUS_CANCELLATION_DENIED	
4	STATUS_INTERNAL_ERROR	
5	STATUS_INVALID_ARGUMENTS	
6	STATUS_RESOURCES_EXCEEDED	
7	STATUS_FILE_TRANSFER_FAILED	
8	STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE	
9	STATUS_UNSUPPORTED_PROTOCOL	
10	STATUS_UPLOAD_FAILED	
11	STATUS_FILE_TRANSFER_FAILED_MULTICAST_GROUP_UNABLE_JOIN	
12	STATUS_FILE_TRANSFER_FAILED_SERVER_CONTACT_FAILED	
13	STATUS_FILE_TRANSFER_FAILED_FILE_ACCESS_FAILED	
14	STATUS_FILE_TRANSFER_FAILED_DOWNLOAD_INCOMPLETE	
15	STATUS_FILE_TRANSFER_FAILED_FILE_CORRUPTED	
16	STATUS_FILE_TRANSFER_FILE_AUTHENTICATION_FAILURE	
19	STATUS_FILE_TRANSFER_WINDOW_EXCEEDED	
20	STATUS_INVALID_UUID_FORMAT	
21	STATUS_UNKNOWN_EXECUTION_ENVIRONMENT	
22	STATUS_DISABLED_EXECUTION_ENVIRONMENT	
23	STATUS_EXECUTION_ENVIRONMENT_MISMATCH	
24	STATUS_DUPLICATE_DEPLOYMENT_UNIT	
25	STATUS_SYSTEM_RESOURCES_EXCEEDED	
26	STATUS_UNKNOWN_DEPLOYMENT_UNIT	
27	STATUS_INVALID_DEPLOYMENT_UNIT_STATE	
28	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_DOWNGRADE_DISALLOWED	
29	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_UPGRADE_DISALLOWED	
30	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_VERSION_EXISTS	
31	STATUS_NOT_FINISHED	
NOTE: See clause 7.4.17 "Resource Type execInstance".		

6.3.4.2.17 m2m:pendingNotification

This is used for the *pendingNotification* attribute of the <subscription> resource.

Table 6.3.4.2.17-1: Interpretation of pendingNotification

Value	Interpretation	Note
1	sendLatest	
2	sendAllPending	
NOTE: See clause 7.4.8 "Resource Type subscription".		

6.3.4.2.18 m2m:notificationContentType

Table 6.3.4.2.18-1: Interpretation of notificationContentType

Value	Interpretation	Note
1	All Attributes	
2	Modified Attributes	
3	ResourceID	
4	Trigger Payload	
NOTE: See clause 7.4.8 "Resource Type subscription".		

6.3.4.2.19 m2m:notificationEventType

Used for ***eventNotificationCriteria*** conditions and in the *notificationEvent* element.

Table 6.3.4.2.19-1: Interpretation of notificationEventType

Value	Interpretation	Note
1	Update_of_Resource	Default
2	Delete_of_Resource	
3	Create_of_Direct_Child_Resource	
4	Delete_of_Direct_Child_Resource	
5	Retrieve_of_Container_Resource_With_No_Child_Resource	Context: A RETRIEVE request targets a subscribed-to <container> resource with the Result Content parameter set to either "child-resources" or "attributes+child-resources". A notification is initiated if the <contentInstance> child resource is obsolete or not present in the targeted parent resource.
6	Trigger_Received_For_AE_Resource	Context: A notification is initiated when a Trigger is Received by a Registrar CSE targeting the AE-ID associated with the <*AE*> resource of a Registree AE.
7	Blocking_Update	

6.3.4.2.20 m2m:status

This is used in [software], [firmware] resources.

Table 6.3.4.2.20-1: Interpretation of status

Value	Interpretation	Note
0	Uninitialized	
1	Successful	
2	Failure	
3	In_Process	
NOTE: See clauses D.2 and D.3 firmware and software management.		

6.3.4.2.21 m2m:batteryStatus

This is used in the [battery] resource.

Table 6.3.4.2.21-1: Interpretation of batteryStatus

Value	Interpretation	Note
1	NORMAL	The battery is operating normally and not on power.
2	CHARGING	The battery is currently charging.
3	CHARGING_COMPLETE	The battery is fully charged and still on power.
4	DAMAGED	The battery has some problem.
5	LOW_BATTERY	The battery is low on charge.
6	NOT_INSTALLED	The battery is not installed.
7	UNKNOWN	The battery information is not available.
NOTE: See clause D.7 battery management.		

6.3.4.2.22 m2m:mgmtDefinition

This is used in the <mgmtObj> resource.

Table 6.3.4.2.22-1: Interpretation of mgmtDefinition

Value	Interpretation	Note
1001	[firmware]	
1002	software	
1003	memory	
1004	areaNwkInfo	
1005	areaNwkDeviceInfo	
1006	battery	
1007	deviceInfo	
1008	deviceCapability	
1009	reboot	
1010	eventLog	
1011	cmdhPolicy	
1012	activeCmdhPolicy	
1013	cmdhDefaults	
1014	cmdhDefEcValue	
1015	cmdhEcDefParamValues	
1016	cmdhLimits	
1017	cmdhNetworkAccessRules	
1018	cmdhNwAccessRule	
1019	cmdhBuffer	
1020	registration	Note 2
1021	dataCollection	Note 2
1022	authenticationProfile	Note 2
1023	myCertFileCred	Note 2
1024	trustAnchorCred	Note 2
1025	MAFClientRegCfg	Note 2
1026	MEFClientRegCfg	Note 2
0	Self-defined	Permits vendor-specific XML schema definition
NOTE 1: See clause 7.4.15 mgmtObj. NOTE 2: These mgmtObj specializations are defined in oneM2M TS-0022 [38].		

6.3.4.2.23 m2m:logTypeId

Used for the *logTypeId* attribute of the [eventLog] Management Resource.

Table 6.3.4.2.23-1: Interpretation of logTypeId

Value	Interpretation	Note
1	System	
2	Security	
3	Event	
4	Trace	
5	Panic	

6.3.4.2.24 m2m:logStatus

Used for the *logStatus* attribute of the [eventLog] Management Resource.

Table 6.3.4.2.24-1: Interpretation of logStatus

Value	Interpretation	Note
1	Started	The logging activity is started.
2	Stopped	The logging activity is stopped.
3	Unknown	The current status of the logging activity is unknown.
4	NotPresent	The log data is not present and the logData attribute shall be ignored.
5	Error	Error conditions for the logging activities, and the logging is stopped.

6.3.4.2.25 m2m:eventType

Used for the *eventType* attribute of the <eventConfig> resource.

Table 6.3.4.2.25-1: Interpretation of eventType

Value	Interpretation	Note
1	DATAOPERATION	
2	STORAGEBASED	
3	TIMERBASED	
NOTE: See clause 7.4.24 "Resource Type eventConfig".		

6.3.4.2.26 m2m:statsRuleStatusType

Used for the *statsRuleStatusType* attribute of the <statsCollect> resource.

Table 6.3.4.2.26-1: Interpretation of statsRuleStatusType

Value	Interpretation	Note
1	ACTIVE	
2	INACTIVE	
NOTE: See clause 7.4.25 "Resource Type statsCollect".		

6.3.4.2.27 m2m:statModelType

Used for the *statModelType* attribute of the <statsCollect> resource.

Table 6.3.4.2.27-1: Interpretation of statModelType

Value	Interpretation	Note
1	EVENTBASED	
NOTE: See clause 7.4.25 "Resource Type statsCollect".		

6.3.4.2.28 m2m:encodingType

Used to describe the encoding type that applies to the *content* attribute of the <contentInstance> resource.

Table 6.3.4.2.28-1: Interpretation of encodingType

Value	Interpretation	Note
0	Plain - no transfer encoding is applied	
1	base64 encoding (see [9]) is applied on string data	
2	base64 encoding (see [9]) is applied on binary data	

6.3.4.2.29 m2m:accessControlOperations

Used in <accessControlPolicy>.

Table 6.3.4.2.29-1: Interpretation of accessControlOperations

Value	Interpretation	Note
1	CREATE	
2	RETRIEVE	
4	UPDATE	
8	DELETE	
16	NOTIFY	
32	DISCOVERY	
NOTE: Combinations of these values are specified by adding them together. For example the value 5 is interpreted as "CREATE and UPDATE".		

6.3.4.2.30 Void

6.3.4.2.31 m2m:filterUsage

Used in m2m:filterCriteria.

Table 6.3.4.2.31-1: Interpretation of filterUsage

Value	Interpretation	Note
1	Discovery Criteria	
2	Conditional Retrieval	This is the default value when the ***filterUsage*** condition is not present in a Retrieve request.
3	IPE On-demand Discovery	

6.3.4.2.32 m2m:notificationTargetPolicyAction

Table 6.3.4.2.32-1: Interpretation of notificationTargetPolicyAction

Value	Interpretation	Note
1	accept request	
2	reject request	
3	seek authorization from subscription originator before responding	
4	inform the subscription originator without taking any action	

6.3.4.2.33 m2m:logicalOperator

Table 6.3.4.2.33-1: Interpretation of logicalOperator

Value	Interpretation	Note
1	AND operation	
2	OR operation	

6.3.4.2.34 m2m:filterOperation

Used in m2m:filterCriteria.

Table 6.3.4.2.34-1: Interpretation of filterOperation

Value	Interpretation	Note
1	Logical AND	This is the default value when the ***filterOperation*** condition is not present in filterCriteria.
2	Logical OR	

6.3.4.2.35 m2m:securityInfoType

Used in m2m:securityInfo.

Table 6.3.4.2.35-1: Interpretation of securityInfoType

Value	Interpretation	Note
1	Dynamic Authorization Request	
2	Dynamic Authorization Response	
3	receiverESPrimRandObject Request	
4	receiverESPrimRandObject Response	
5	ESPrim Object	
6	ESCertKE Message	
7	Dynamic Authorization Relationship Mapping Request	
8	Dynamic Authorization Relationship Mapping Response	

6.3.4.2.36 m2m:allJoynDirection

Used for the *direction* attribute of the [allJoynApp] resource.

Table 6.3.4.2.36-1: Interpretation of allJoynDirection

Value	Interpretation	Note
1	AllJoyn_to_oneM2M	
2	oneM2M_to_AllJoyn	
NOTE: See clause J.6 "Resource type [allJoynApp]".		

6.3.4.2.37 m2m:contentFilterSyntax

Used for the ***contentFilterSyntax*** element in the ***Filter Criteria*** primitive parameter.

Table 6.3.4.2.37-1: Interpretation of contentFilterSyntax

Value	Interpretation	Note
1	JSON_PATH_SYNTAX	'jsonpath' query syntax. See clause K.2 of oneM2M TS-0001 [6] for information about the jsonpath syntax.

6.3.4.2.38 m2m:contentSecurity

Used in m2m:contentInfo.

Table 6.3.4.2.38-1: Interpretation of contentSecurity

Value	Interpretation	Note
0	ESData has not been applied to the content data.	
1	ESData using JWE and/or JWS with Compact Serialization has been applied to the content data with no subsequent transfer encoding. See note 2.	See note 3
2	ESData using JWE and/or JWS with JSON Serialization has been applied to the content data with no subsequent transfer encoding.	See note 4
3	ESData using JWE and/or JWS with JSON Serialization has been applied to the content data and subsequent base64 encoding (see [9]) has been applied.	See note 4
4	ESData using XML encryption and/or XML-Signature has been applied to the content data has been applied with no subsequent transfer encoding.	See note 5
5	ESData using XML encryption and/or XML-Signature has been applied to the content data and subsequent base64 encoding (see [9]) has been applied.	See note 5

NOTE 1: See oneM2M TS-0003 [7] for details on these security protocols.
NOTE 2: JWE and/or JWS with Compact Serialization is almost entirely base64url encoded by default - see m2m:e2eCompactJWS and m2m:e2eCompactJWE in Table 6.3.3-1. Consequently, there is no option for additional base64 encoding of JWE and/or JWS with Compact Serialization.
NOTE 3: The ESData envelope (see oneM2M TS-0003 [7]) in this case has media type application/jose.
NOTE 4: The ESData envelope in this case has media type application/jose+json.
NOTE 5: The ESData envelope in this case has media type application/xenc+xml or application/xml (the latter is the media type for XML Signature).

6.3.4.2.39 m2m:suid

Used in m2m:e2eSecInfo and other security features in oneM2M TS-0003 [7].

NOTE: This enumeration is the concatenation of two identifiers. The first identifier identifies the type of credential (such as pre-provisioned symmetric key, symmetric key provisioned via a RSPF, or symmetric key distributed via MAF and certificate) and the intended scope within which the credential is to be used (such as shared with an MEF, shared with an MAF, use in SAEF, use in ESPrim, use in authentication encryption in ESData, or use in signature only in ESData).

Table 6.3.4.2.39-1: Interpretation of suid

Value	Interpretation	Note
10	A pre-provisioned symmetric key intended to be shared with a MEF	
11	A pre-provisioned symmetric key intended to be shared with a MAF	
12	A pre-provisioned symmetric key intended for use in a Security Associated Establishment Framework (SAEF)	
13	A pre-provisioned symmetric key intended for use in End-to-End Security of Primitives (ESPrim)	
14	A pre-provisioned symmetric key intended for use with authenticated encryption in the Encryption-only or Nested Sign-then-Encrypt End-to-End Security of Data (ESData) Data classes	
15	A pre-provisioned symmetric key intended for use in Signature-only ESData Security Class	
21	A symmetric key, provisioned via a Remote Security Provisioning Framework (RSPF), and intended to be shared with a MAF	
22	A symmetric key, provisioned via a RSPF, and intended for use in a SAEF	
23	A symmetric key, provisioned via a RSPF, and intended for use in ESPrim	
24	A symmetric key, provisioned via a RSPF, and intended for use with authenticated encryption in the Encryption-only or Nested Sign-then-Encrypt ESData) Data classes	
25	A symmetric key, provisioned via a RSPF, and intended for use in Signature-only ESData Security Class	
32	A MAF-distributed symmetric key intended for use in a SAEF	
33	A MAF-distributed symmetric key intended for use in ESPrim	
34	A MAF-distributed symmetric key intended for use with authenticated encryption in the Encryption-only or Nested Sign-then-Encrypt ESData Data classes	
35	A MAF-distributed symmetric key intended for use in Signature-only ESData Security Class	
40	A certificate intended to be shared with a MEF	
41	A certificate intended to be shared with a MAF	
42	A certificate intended for use in a Security Associated Establishment Framework (SAEF)	
43	A certificate intended for use in End-to-End Security of Primitives (ESPrim)	
44	A certificate intended for use with authenticated encryption in the Encryption-only or Nested Sign-then-Encrypt End-to-End Security of Data (ESData) Data classes	
45	A certificate intended for use in Signature-only ESData Security Class	
NOTE: See oneM2M TS-0003 [7] for further detail.		

6.3.4.2.40 m2m:esprimKeyGenAlgID

Used in m2m:receiverESPrimRandObject and m2m:originatorESPrimRandObject.

Table 6.3.4.2.40-1: Interpretation of esprimKeyGenAlgID

Value	Interpretation	Note
1	HMAC-SHA256	See oneM2M TS-0003 [7]
NOTE: Further sessionESPrimKey Key Generation Algorithms are anticipated to be added in the future.		

6.3.4.2.41 m2m:esprimProtocolAndAlgID

Used in m2m:receiveESPrimRandObject and m2m:originatorESPrimRandObject.

NOTE: This enumeration is the concatenation of two identifiers. The most significant numeral identifies an object security technology (that is, a protocol) such as JSON Web Encryption (JWE) Compact Representation [i.8]. Further protocols can be supported in the future. The least significant numeral identifies an Authenticated Encryption option for that object security technology.

Table 6.3.4.2.41-1: Interpretation of esprimProtocolAndAlgID

Value	Interpretation	Notes
10	JWE Compact Serialization with "enc" = "A128GCM"	AES GCM using 128-bit key
11	JWE Compact Serialization with "enc" = "A192GCM"	AES GCM using 192-bit key
12	JWE Compact Serialization with "enc" = "A256GCM"	AES GCM using 256-bit key
NOTE 1: JWE Compact Serialization for ESPrim is specified in clause 8.4.3.2 of oneM2M TS-0003 [7]. NOTE 2: JWE authentication encryption using AES GCM is specified in [i.8] and [i.9].		

6.3.4.2.42 Void

6.3.4.2.43 m2m:stationaryIndication

Used for the *stationaryIndication* element of m2m:activityPattern.

Table 6.3.4.2.43-1: Interpretation of stationaryIndication

Value	Interpretation	Note
1	Stationary	
2	Mobile (Moving)	

6.3.4.2.44 m2m:contentStatus

Used for the ***Content Status*** response primitive parameter.

Table 6.3.4.2.44-1: Interpretation of contentStatus

Value	Interpretation	Note
1	PARTIAL_CONTENT	
2	FULL_CONTENT	
NOTE: See clause 6.4.2 "Response primitive parameter data types".		

6.3.4.2.45 m2m:networkAction

Used for the *networkAction* element in the m2m:backOffParameters.

Table 6.3.4.2.45-1: Interpretation of networkAction

Value	Interpretation	Note
1	cellular-registration	
2	cellular-attach	
3	cellular-pdpctxact	
4	cellular-sms	
5	default	

6.3.4.2.46 m2m:locationInformationType

Used for the *locationInformationType* attribute of the <locationPolicy> resource.

Table 6.3.4.2.46-1: Interpretation of m2m:locationInformationType

Value	Interpretation	Note
1	Position fix	
2	Geofence event	

6.3.4.2.47 m2m:geofenceEventCriteria

Used for the *geofenceEventCriteria* attribute of the <locationPolicy> resource.

Table 6.3.4.2.47-1: Interpretation of m2m:geofenceEventCriteria

Value	Interpretation	Note
1	Entering	
2	Leaving	
3	Inside	
4	Outside	

6.3.4.2.48 m2m:semanticFormat

Used in the <semanticDescriptor> and <ontology> resources.

Table 6.3.4.2.48-1: Interpretation of semanticFormat

Value	Interpretation	Note
1	IRI	See [11]. This shall not be used for the *descriptorRepresentation* of a <semanticDescriptor> resource.
2	File format: Functional-style	See [44]
3	File format: OWL/XML	See [45]
4	File format: RDF/XML	See [46] and [34]
5	File format: RDF/Turtle	See [46] and [50]
6	File format: Manchester	See [48]
7	File format: JSON-LD	See [49]

6.3.4.2.49 m2m:triggerPurpose

Used in defining trigger purpose in a trigger payload.

Table 6.3.4.2.49-1: Interpretation of triggerPurpose

Value	Interpretation	Note
1	establishConnection	
2	registrationRequest	
3	executeCRUD	
4	enrolmentRequest	

6.3.4.2.50 m2m:serializationType

Used by a trigger originator to specify its supported types of serialization.

Table 6.3.4.2.50-1: Interpretation of serializationType

Value	Interpretation	Note
1	XML	
2	JSON	
4	CBOR	
NOTE: Combinations of these values are specified by adding them together. For example the value 3 is interpreted as "XML and JSON".		

6.3.4.2.51 m2m:authorizationDecision

Used for the *decision* attribute of the <authorizationDecision> resource.

Table 6.3.4.2.51-1: Interpretation of authorizationDecision

Value	Interpretation	Note
1	PERMIT	The requested access is permitted.
2	DENY	The requested access is denied.
NOTE: See clause 7.4.43 "Resource Type authorizationDecision".		

6.3.4.2.52 m2m:authorizationStatus

Used for the *status* attribute of the <authorizationDecision> resource.

Table 6.3.4.2.52-1: Interpretation of authorizationStatus

Value	Interpretation	Note
1	OK	It indicates the access control policy evaluation, access control policy retrieval, or access control information retrieval procedure is successful.
2	NOT_APPLICABLE	It indicates there is no applicable access control policy or requested access control information.
3	MISSING_ATTRIBUTE	It indicates that some access control information required to make an access control decision is not available, e.g. roles or tokens.
4	SYNTAX_ERROR	It indicates there is a syntax error in access control information or an access control policy, e.g. invalid tokens or access control rules.
5	PROCESSING_ERROR	It indicates an error occurred during the access control policy evaluation, access control policy retrieval or access control information retrieval procedure.
NOTE: See clause 7.4.43 "Resource Type authorizationDecision", clause 7.4.44 "Resource Type authorizationPolicy" and clause 7.4.45 "Resource Type authorizationInformation".		

6.3.4.2.53 m2m:acpCombiningAlgorithm

Used for the *combiningAlgorithm* attribute of the <authorizationPolicy> resource.

Table 6.3.4.2.53-1: Interpretation of acpCombiningAlgorithm

Value	Interpretation	Note
1	PERMIT_OVERRIDES	When evaluating multiple access control policies, if a single "Permit" result is encountered, then the combined result is "Permit".
NOTE: See clause 7.4.44 "Resource Type authorizationPolicy".		

6.3.4.2.54 m2m:mashupMemberStoreType

Used for ***memberStoreType*** attribute of <semanticMashupInstance> resource.

Table 6.3.4.2.54-1: Interpretation of mashupMemberStoreType

Value	Interpretation	Note
1	URI_ONLY	
2	URI_AND_VALUE	
NOTE: See clause 7.4.50 "Resource Type semanticMashupInstance".		

6.3.4.2.55 m2m:mashupResultGenType

Used for ***resultGenType*** attribute of <semanticMashupInstance> resource.

Table 6.3.4.2.55-1: Interpretation of authorizationDecision mashupResultGenType

Value	Interpretation	Note
1	WHEN_SMI_IS_CREATED	
2	WHEN_MR_REQUESTS	
3	PERIODICALLY	
4	WHEN_A_MASHUP_MEMBER_IS_UPDATED	
NOTE: See clause 7.4.50 "Resource Type semanticMashupInstance".		

6.3.4.2.56 m2m:locationUpdateEventCriteria

Used for locationUpdateEventCriteria attribute of <locationPolicy> resource.

Table 6.3.4.2.56-1: Interpretation of locationUpdateEventCriteria

Value	Interpretation	Note
0	Location_Change	

6.3.4.2.57 m2m:AERegistrationStatus

Used for *registrationStatus* attribute in <AE> resource.

Table 6.3.4.2.57-1: Interpretation of AERegistrationStatus

Value	Interpretation	Note
1	ACTIVE	
2	INACTIVE	
NOTE: See clause 7.4.5 "Resource Type AE".		

6.3.4.2.58 m2m:multicastCapability

Used for the *multicastCapability* attribute of the <remoteCSE> resource and the *multicastType* element in the Multicast Group Information data object.

Table 6.3.4.2.58-1: Interpretation of multicastCapability

Value	Interpretation	Note
1	MBMS	
2	IP	

6.3.4.2.59 m2m:sessionState

Used for the *sessionState* attribute of the <multimediaSession> resource.

Table 6.3.4.2.59-1: Interpretation of sessionState

Value	Interpretation	Note
1	OFFLINE	This is default when the attribute is not given during the creation.
2	ONLINE	
NOTE: See clause 7.4.56 "Resource Type multimediaSession".		

6.3.4.2.60 m2m:triggerStatus

Used for the *triggerStatus* attribute of the <triggerRequest> resource.

Table 6.3.4.2.60-1: Interpretation of triggerStatus

Value	Interpretation	Note
1	PROCESSING	
2	ERROR_NSE_NOT_FOUND	
3	TRIGGER_SUBMITTED	
4	TRIGGER_DELIVERED	
5	TRIGGER_EXPIRED	
6	TRIGGER_FAILED	

6.3.4.2.61 m2m:timeWindowType

Used for the *timeWindowType* attribute of the <crossResourceSubscription> resource.

Table 6.3.4.2.61-1: Interpretation of timeWindowType

Value	Interpretation	Note
1	PERIODICWINDOW	
2	SLIDINGWINDOW	
NOTE: See clause 7.4.58 "Resource Type crossResourceSubscription".		

6.3.4.2.62 m2m:transferSelectionGuidance

Used for the *transferSelectionGuidance* attribute of the <backgroundDataTransfer> resource. Specified by an originator to provide guidance, together with local policies, on how to choose a Background Data Transfer policy if multiple policies are provided to the IN-CSE by the SCEF.

Table 6.3.4.2.62-1: Interpretation of transferSelectionGuidance

Value	Interpretation	Note
1	lowestCost	
2	bestDataRates	

6.3.4.2.63 m2m:transactionMode

This is used for the *transactionMode* attribute of the <transactionMgmt> resource to define whether the Hosting CSE or the creator of a <transactionMgmt> resource is responsible for controlling the execution of the transaction.

Table 6.3.4.2.63-1: Interpretation of transactionMode

Value	Interpretation	Note
1	CSE_CONTROLLED	This is the default
2	CREATOR_CONTROLLED	

6.3.4.2.64 m2m:transactionControl

This is used for the *transactionControl* attribute of the <transactionMgmt> and <transaction> resources to control the state of a transaction.

Table 6.3.4.2.64-1: Interpretation of transactionControl

Value	Interpretation	Note
1	INITIAL	
2	LOCK	
3	EXECUTE	
4	COMMIT	
5	ABORT	

6.3.4.2.65 m2m:transactionState

This is used for the *transactionState* attribute of the <transactionMgmt> and <transaction> resources to monitor the state of a transaction.

Table 6.3.4.2.65-1: Interpretation of transactionState

Value	Interpretation	Note
1	INITIAL	
2	LOCKED	
3	EXECUTED	
4	COMMITTED	
5	ERROR	
6	ABORTED	

6.3.4.2.66 m2m:transactionLockType

This is used for the *transactionLockType* attribute of the <transactionMgmt> and <transaction> resources to configure the type of lock that is required on the targeted resource in order to perform the transaction.

Table 6.3.4.2.66-1: Interpretation of transactionLockType

Value	Interpretation	Note
1	BLOCK_ALL	
2	ALLOW_RETRIEVES	

6.3.4.2.67 m2m:transactionMgmtHandling

This is used for the *transactionMgmtHandling* attribute of the <transactionMgmt> resource to configure whether to persist or delete the <transactionMgmt> resource after its completion.

Table 6.3.4.2.67-1: Interpretation of transactionMgmtHandling

Value	Interpretation	Note
1	DELETE	
2	PERSIST	

6.3.5 Complex data types

6.3.5.1 Introduction

The present clause defines structured information for specific use in oneM2M protocol. These types are defined to be xs:sequence complex types, unless specified otherwise. XML Schema data type definitions for these data types can be found in the XSD file called CDT-commonTypes-v3_11_0.xsd.In addition, each oneM2M resource has a corresponding complex data type. These are described in clause 6.5.

6.3.5.2 m2m:deliveryMetaData

Used for the *deliveryMetaData* attribute of the <delivery> resource.

Table 6.3.5.2-1: Type Definition of m2m:deliveryMetadata

Element Path	Element Data Type	Multiplicity	Note
tracingOption	xs:boolean	1	
tracingInfo	m2m:listOfM2MID	0..1	

6.3.5.3 m2m:aggregatedRequest

Used for the *aggregatedRequest* attribute of the <delivery> resource.

Table 6.3.5.3-1: Type Definition of m2m:aggregatedRequest

Element Path	Element Data Type	Multiplicity	Note
request	(anonymous)	1..n	
request/operation	m2m:operation	1	See clause 6.3.4.2.5
request/to	xs:anyURI	1	
request/from	m2m:ID	1	See clause 6.3.3
request/requestIdentifier	m2m:requestID	1	See clause 6.3.3
request/primitiveContent	m2m:primitiveContent	0..1	See clause 6.3.5.5
request/metaInformation	m2m:metaInformation	0..1	See clause 6.3.5.4

6.3.5.4 m2m:metaInformation

Used for the *metaInformation* attribute of the <request> resource, and in the m2m:aggregatedRequest data type.

Table 6.3.5.4-1: Type Definition of m2m:metaInformation

Element Path	Element Data Type	Multiplicity	Note
resourceType	m2m:resourceType	0..1	See clause 6.3.4.2.1
originatingTimestamp	m2m:timestamp	0..1	
requestExpirationTimestamp	m2m:absRelTimestamp	0..1	
resultExpirationTimestamp	m2m:absRelTimestamp	0..1	
operationExecutionTime	m2m:absRelTimestamp	0..1	
responseType	m2m:responseTypeInfo	0..1	See clause 6.3.4.2.6
resultPersistence	m2m:absRelTimestamp	0..1	
resultContent	m2m:resultContent	0..1	See clause 6.3.4.2.7
eventCategory	m2m:eventCat	0..1	See clause 6.3.3
deiveryAggregation	xs:boolean	0..1	
groupRequestIdentifier	xs:string	0..1	
filterCriteria	m2m:filterCriteria	0..1	See clause 6.3.5.8
desiredIdentifierResultType	m2m:desIdResType	0..1	See clause 6.3.4.2.8
roleIDs	List of m2m:roleID	0..1	
tokenRequestIndicator	xs:boolean	0..1	
tokens	List of m2m:dynAuthJWT	0..1	
tokenIDs	List of m2m:tokenID	0..1	
localTokenIDs	List of xs:NCName	0..1	
groupRequestTargetMembers	List of xs:anyURI	0..1	
authorSignIndicator	xs:boolean	0..1	
authorSigns	m2m:signatureList	0..1	
authorRelIndicator	xs:boolean	0..1	
semanticQueryIndicator	xs:boolean	0..1	
ReleaseVersionIndicator	m2m:releaseVersion	1	
vendorInformation	xs:string	0..1	

6.3.5.5 m2m:primitiveContent

Used for the ***Content*** parameter in request/response primitives and the *content* attribute of the <request> resource.

See clauses 7.2.1.1 and 7.2.1.2.

6.3.5.6 m2m:batchNotify

Used for the *batchNotify* attribute of the <subscription> resource and the *notifyAggregation* attribute of the <group> resource.

Table 6.3.5.6-1: Type Definition of m2m:batchNotify

Element Path	Element Data Type	Multiplicity	Note
number	xs:nonNegativeInteger	0..1	
duration	xs:duration	0..1	If the duration is not given by the Originator, the Hosting CSE shall set this with the default duration value as given by the M2M Service Provider.

6.3.5.7 m2m:eventNotificationCriteria

Used for the *eventNotificationCriteria* attribute of the <subscription> resource.

Table 6.3.5.7-1:Type Definition of m2m:eventNotificationCriteria

Element Path	Element Data Type	Multiplicity	Note
createdBefore	m2m:timestamp	0..1	
createdAfter	m2m:timestamp	0..1	
modifiedSince	m2m:timestamp	0..1	
unmodifiedSince	m2m:timestamp	0..1	
stateTagSmaller	xs:positiveInteger	0..1	
stateTagBigger	xs:nonNegativeInteger	0..1	
expireBefore	m2m:timestamp	0..1	
expireAfter	m2m:timestamp	0..1	
sizeAbove	xs:nonNegativeInteger	0..1	
sizeBelow	xs:positiveInteger	0..1	
operationMonitor	m2m:operationMonitor	0..n	
attribute	m2m:attributeList	0..1	
notificationEventType	m2m:notificationEventType	0..6	
childResourceType	list of m2m:resourceType	0..1	
missingData	m2m:missingData	0..1	
filterOperation	m2m:filterOperation	0..1	

6.3.5.8 m2m:filterCriteria

Used indirectly in the <request> resource and for the ***Filter Criteria*** parameter in a request.

Table 6.3.5.8-1: Type Definition of m2m:filterCriteria

Element Path	Element Data Type	Multiplicity	Note
createdBefore	m2m:timestamp	0..1	
createdAfter	m2m:timestamp	0..1	
modifiedSince	m2m:timestamp	0..1	
unmodifiedSince	m2m:timestamp	0..1	
stateTagSmaller	xs:positiveInteger	0..1	
stateTagBigger	xs:nonNegativeInteger	0..1	
expireBefore	m2m:timestamp	0..1	
expireAfter	m2m:timestamp	0..1	
labels	m2m:labels	0..1	
childLabels	m2m:labels	0..1	
parentLabels	m2m:labels	0..1	
labelsQuery	xs:string	0..1	
resourceType	list of m2m:resourceType	0..1	
childResourceType	list of m2m:resourceType	0..1	
parentResourceType	list of m2m:resourceType	0..1	
sizeAbove	xs:nonNegativeInteger	0..1	
sizeBelow	xs:positiveInteger	0..1	
contentType	m2m:typeOfContent	0..n	
attribute	m2m:attribute	0..n	
childAttribute	m2m:attribute	0..n	
parentAttribute	m2m:attribute	0..n	
filterUsage	m2m:filterUsage	0..1	
limit	xs:nonNegativeInteger	0..1	
semanticsFilter	m2m:sparql	0..n	
filterOperation	m2m:filterOperation	0..1	
contentFilterSyntax	m2m:contentFilterSyntax	0..1	
contentFilterQuery	xs:string	0..1	
level	xs:positiveInteger	0..1	
offset	xs:positiveInteger	0..1	
applyRelativePath	xs:anyURI	0..1	

6.3.5.9 m2m:attribute

Used in m2m:filterCriteria.

Table 6.3.5.9-1: Type Definition of m2m:attribute

Element Path	Element Data Type	Multiplicity	Note
name	xs:NCName	1	
value	xs:anyType	1	

6.3.5.10 Void

6.3.5.11 m2m:scheduleEntries

Table 6.3.5.11-1: Type Definition of m2m:scheduleEntries

Element Path	Element Data Type	Multiplicity	Note
scheduleEntry	m2m:scheduleEntry	1..n	

6.3.5.12 m2m:aggregatedNotification

Used in the Notification Data Object.

Table 6.3.5.12-1: Type Definition of m2m:aggregatedNotification

Element Path	Element Data Type	Multiplicity	Note
notification	m2m:notification	1..n	

6.3.5.13 m2m:notification

Table 6.3.5.13-1: Type Definition of m2m:notification

Element Path	Element Data Type	Multiplicity	Note
notificationEvent	(anonymous)	0..1	
notificationEvent/representation	m2m:representation	0..1	See Table 6.3.5.62-1.
notificationEvent/operationMonitor	(anonymous)	0..1	
notificationEvent/operationMonitor/operation	m2m:operation	1	m2m:operation This element shall only be present if the operationMonitor parent element is present. Otherwise it shall not.
notificationEvent/operationMonitor/originator	m2m:ID	1	m2m:ID This element shall only be present if the operationMonitor parent element is present. Otherwise it shall not.
notificationEvent/notificationEventType	m2m:notificationEventType	1	This element shall only be present if the notificationEvent parent element is present. Otherwise it shall not.
verificationRequest	xs:boolean	0..1	
subscriptionDeletion	xs:boolean	0..1	
subscriptionReference	xs:anyURI	1	
creator	m2m:ID	0..1	
notificationForwardingURI	xs:anyURI	0..1	
notificationTarget	m2m:ID	0..1	
targetRemovalRequest	xs:boolean	0..1	
targetRemovalAllowance	xs:boolean	0..1	
IPEDiscoveryRequest	(anonymous)	0..1	
IPEDiscoveryRequest/originator	m2m:ID	1	This element shall only be present if the IPEDiscoveryRequest parent element is present. Otherwise it shall not.
IPEDiscoveryRequest/filterCriteria	m2m:filterCriteria	1	This element shall only be present if the IPEDiscoveryRequest parent element is present. Otherwise it shall not.
AERegistrationPointChange	xs:boolean	0..1	
AEReferenceIDChange	xs:boolean	0..1	
trackingID1	m2m:ID	1	
trackingID2	m2m:ID	1	

6.3.5.14 m2m:actionStatus

Table 6.3.5.14-1: Type Definition of m2m:actionStatus

Element Path	Element Data Type	Multiplicity	Note
action	xs:anyURI	0..1	Reference to the action (represented by a resource attribute) being performed
status	m2m:status	0..1	Indicates the status of the operation is successful, failure or in process. See Table 6.3.4.2.20-1

6.3.5.15 m2m:anyArgType

Table 6.3.5.15-1: Type Definition of m2m:anyArgType

Element Path	Element Data Type	Multiplicity	Note
name	xs:NCName	1	
value	xs:anyType	1	

6.3.5.16 m2m:resetArgsType

Table 6.3.5.16-1: Type Definition of m2m:resetArgsType

Element Path	Element Data Type	Multiplicity	Note
anyArg	m2m:anyArgType	0..n	

6.3.5.17 m2m:rebootArgsType

Table 6.3.5.17-1: Type Definition of m2m:rebootArgsType

Element Path	Element Data Type	Multiplicity	Note
anyArg	m2m:anyArgType	0..n	

6.3.5.18 m2m:uploadArgsType

Table 6.3.5.18-1: Type Definition of m2m:uploadArgsType

Element Path	Element Data Type	Multiplicity	Note
fileType	xs:string	1	
URL	xs:anyURI	1	
username	xs:string	1	
password	xs:string	1	
anyArg	m2m:anyArgType	0..n	

6.3.5.19 m2m:downloadArgsType

Table 6.3.5.19-1: Type Definition of m2m:downloadArgsType

Element Path	Element Data Type	Multiplicity	Note
fileType	xs:string	1	
URL	xs:anyURI	1	
username	xs:string	1	
password	xs:string	1	
filesize	xs:positiveInteger	1	
targetFile	xs:string	1	
delaySeconds	xs:positiveInteger	1	
successURL	xs:anyURI	1	
startTime	m2m:timestamp	1	
completeTime	m2m:timestamp	1	
anyArg	m2m:anyArgType	0..n	

6.3.5.20 m2m:softwareInstallArgsType

Table 6.3.5.20-1: Type Definition of m2m:softwareInstallArgsType

Element Path	Element Data Type	Multiplicity	Note
URL	xs:anyURI	1	
UUID	xs:string	1	
username	xs:string	1	
password	xs:string	1	
executionEnvRef	xs:string	1	
anyArg	m2m:anyArgType	0..n	

6.3.5.21 m2m:softwareUpdateArgsType

Table 6.3.5.21-1: Type Definition of m2m:softwareUpdateArgsType

Element Path	Element Data Type	Multiplicity	Note
UUID	xs:string	1	
version	xs:string	1	
URL	xs:anyURI	1	
username	xs:string	1	
password	xs:string	1	
executionEnvRef	xs:string	1	
anyArg	m2m:anyArgType	0..n	

6.3.5.22 m2m:softwareUninstallArgsType

Table 6.3.5.22-1: Type Definition of m2m:softwareUninstallArgsType

Element Path	Element Data Type	Multiplicity	Note
UUID	xs:string	1	
version	xs:string	1	
executionEnvRef	xs:string	1	
anyType	m2m:anyArgType	0..n	

6.3.5.23 m2m:execReqArgsListType

Table 6.3.5.23-1: Type Definition of m2m:execReqArgsListType

Element Path	Element Data Type	Multiplicity	Note
reset	m2m:resetArgsType	0..n	
reboot	m2m:rebootArgsType	0..n	
upload	m2m:uploadArgsType	0..n	
download	m2m:downloadArgsType	0..n	
softwareInstall	m2m:softwareInstallArgsType	0..n	
softwareUpdate	m2m:softwareUpdateType	0..n	
softwareUninstall	m2m:softwareUninstallArgsType	0..n	

This type is an xs:choice. It shall contain elements from no more than one row listed in Table 6.3.5.23-1.

6.3.5.24 m2m:mgmtLinkRef

Table 6.3.5.24-1: Type Definition of m2m:mgmtLinkRef

Element Path	Element Data Type	Multiplicity	Note
(base content)	xs:anyURI	1	URI (of type xs:anyURI) with name and type attributes
@name	m2m:resourceName	1	The name attribute represents the name of the referenced resource instance
@type	m2m:mgmtDefinition	1	The type attribute is restricted to the allowed specializations of resource type <mgmtObj>

In Table 6.3.5.24-1, names of XML schema attributes are prefixed with a "@" character to differentiate these from Resource attribute names. The "@" character is not part of the actual attribute name.

6.3.5.25 m2m:resourceWrapper

This data type is used for the m2m:resource Global Element of the ***Content*** primitive parameter as defined in clause 7.5.2. It allows insertion of Global Elements prefixed with the m2m: namespace identifier, or of Global Elements defined in another namespace.

Table 6.3.5.25-1: Type Definition of m2m:resourceWrapper

Element Path	Element Data Type	Multiplicity	Note
m2m:<resourceType> {other namespace identifier}:<resourceType>	(anonymous)	1	A representation of a Resource with specific type as described in clause 7.4
URI	xs:anyURI	1	Hierarchical URI of the resource

6.3.5.26 m2m:setOfAcrs

Table 6.3.5.26-1: Type Definition of m2m:setOfAcrs

Element Path	Element Data Type	Multiplicity	Note
accessControlRules	m2m:accessControlRule	0..n	Data type of privileges and selfPrivileges attributes

6.3.5.27 m2m:accessControlRule

Table 6.3.5.27-1: Type Definition of m2m:accessControlRule

Element Path	Element Data Type	Multiplicity	Note
accessControlOriginators	list of xs:anyURI	1	See clause 7.3.3.15 for the detail
accessControlOperations	m2m:accessControlOperations	1	
accessControlContexts		0..n	
accessControlContexts/accessControlWindow	m2m:scheduleEntry	0..n	
accessControlContexts/accessControlIpAddresses		0..1	
accessControlContexts/accessControlIpAddresses/ipv4Addresses	list of m2m:ipv4	0..1	List of IPv4 addresses
accessControlContexts/accessControlIpAddresses/ipv6Addresses	list of m2m:ipv6	0..1	List of IPv6 addresses
accessControlContexts/accessControlLocationRegions	m2m:locationRegion	0..1	
accessControlAuthenticationFlag	xs:boolean	0..1	
accessControlObjectDetails		0..n	
accessControlObjectDetails/resourceType	m2m:resourceType	0..1	resourceType identifier of the targeted parent resource
accessControlObjectDetails/specializationType	m2m:specializationType	0..1	This could be a containerDefinition or mgmtDefinition
accessControlObjectDetails/childResourceType	list of m2m:resourceType	1	
NOTE: Some of the above elements are defined in clause 9.6.2 of oneM2M TS-0001 [6] with slightly different names as follows (name in parenthesis used in oneM2M TS-0001 [6]): accessControlWindow (accessControlTimeWindow), accessControlIpAddresses (accessControlIpAddress), specializationID (specialization).			

The accessControlContexts/accessControlIpAddresses element may include either the ipv4Addresses element, ipv6Addresses element, or both elements.

Each individual IPv4 address of data type m2m:ipv4 in the list of IPv4 addresses is represented in dotted-decimal notation with optional Classless Inter-Domain Routing (CIDR) suffix in accordance with IETF RFC 4632 [29]. Each individual IPv6 address of data type m2m:ipv6 in the list of IPv6 addresses is represented in colon separated groups of hexadecimal digits with optional network prefix in accordance with IETF RFC 5952 [30]. Example IPv4 and IPv6 addresses which comply with data types m2m:ipv4 and m2m:ipv6, respectively, are given in Table 6.3.2-1. If the accessControlAuthenticationFlag element is not present, then the value is assumed to be false.

6.3.5.28 m2m:locationRegion

Table 6.3.5.28-1: Type Definition of m2m:locationRegion

Element Path	Element Data Type	Multiplicity	Note
circRegion	list of 3 xs:float	0..1	The values represent latitude (+/-90 degrees), longitude (+/-180 degrees), and radius (metres)
countryCode	list of m2m:countryCode	0..1	

This is an xs:choice. A locationRegion shall contain either:

1) A countryCode element, in which case circRegion shall not appear, or

2) A circRegion element, in which case countryCode shall not appear.

6.3.5.29 m2m:childResourceRef

Table 6.3.5.29-1: Type Definition of m2m:childResourceRef

Element Path	Element Data Type	Multiplicity	Note
(base content)	xs:anyURI	1	URI of the child resource using the addressing format specified by the ***Desired Identifier Result Type*** request attribute.
@name	m2m:resourceName	1	*Gives the* **name** *of the child resource pointed to by the URI*
@type	m2m:resourceType	1	*Gives the resourceType of the child resource pointed to by the URI*
@specializationID	xs:anyURI	0..1	*Gives resource type specialization of the child resource pointed to by the URI in case @type represents a <flexContainer>*

In Table 6.3.5.29-1, names of XML schema attributes are prefixed with a "@" character to differentiate these from Resource attribute names. The "@" character is not part of the actual attribute name.

6.3.5.30 m2m:responseTypeInfo

Table 6.3.5.30-1: Type Definition of m2m:responseTypeInfo

Element Path	Element Data Type	Multiplicity	Note
responseTypeValue	m2m:responseType	1	See clause 6.3.4.2.6
notificationURI	list of xs:anyURI	0..1	This element may be included only when the responseType is set to "2" (nonBlockingRequest Asynch). Empty list in this element shall be allowed. See clause 7.5.1.2.5

6.3.5.31 m2m:rateLimit

Used in <subscription>.

Table 6.3.5.31-1: Type Definition of m2m:rateLimit

Element Path	Element Data Type	Multiplicity	Note
maxNrOfNotify	xs:nonNegativeInteger	0..1	
timeWindow	xs:duration	0..1	

6.3.5.32 m2m:operationResult

Used for the *operationResult* attribute of the <request> resource.

NOTE: This data type corresponds to the sequence of elements in the response primitive defined in clause 6.4.2.

Table 6.3.5.32-1: Type Definition of m2m:operationResult

Element Path	Element Data Type	Multiplicity	Note
responseStatusCode	m2m:responseStatusCode	1	See clause 6.3.4.2.9
requestIdentifier	m2m:requestID	1	See clause 6.3.3
primitiveContent	m2m:primitiveContent	0..1	See clause 6.3.5.5
to	xs:anyURI	0..1	
from	m2m:ID	0..1	See clause 6.3.3
originatingTimestamp	m2m:timestamp	0..1	
resultExpirationTimestamp	m2m:absRelTimestamp	0..1	
eventCategory	m2m:eventCat	0..1	See clause 6.3.3
contentStatus	m2m:contentStatus	0..1	See clause 6.3.4.2.44
contentOffset	xs:positiveInteger	0..1	

6.3.5.33 m2m:aggregatedResponse

Used when aggregating responses by a group.

Table 6.3.5.33-1: Type Definition of m2m:aggregatedResponse

Element Path	Element Data Type	Multiplicity	Note
resourceID	xs:anyURI	0..1	Reference to the <request> resource that can be used to retrieve the remaining member responses
m2m:responsePrimitive	See Table 6.4.2-1 for detail	0..n	See note
NOTE: The element name shall contain the namespace prefix.			

6.3.5.34 m2m:mgmtResource

Used This data type is used as base type of all <*mgmtObj*> specializations specified in Annex D. It includes the attributes commonly used by all <*mgmtObj*> resource type specializations.

It consists of the common attributes included in m2m:announceableResource as defined Table 6.3.6-2 and the additional common attributes shared by all <*mgmtObj*> specializations shown in Table 6.3.5.34-1.

Table 6.3.5.34-1: Type Definition of m2m:mgmtResource

Element Path	Element Data Type	Multiplicity	Note
@resourceName	m2m:resourceName	1	common attributes as defined in m2m:announceable Resource, see Table 6.3.6-2
resourceType	m2m:resourceType	1	
resourceID	m2m:ID	1	
parentID	m2m:nhURI	1	
creationTime	m2m:timestamp	1	
lastModifiedTime	m2m:timestamp	1	
labels	m2m:labels	0..1	
accessControlPolicyIDs	m2m:acpType	0..1	
expirationTime	m2m:timestamp	1	
dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
announceTo	list of xs:anyURI	0..1	
announcedAttribute	list of xs:NCName	0..1	
mgmtDefinition	m2m:mgmtDefinition	1	
objectIDs	m2m:listOfURIs	0..1	
objectPaths	m2m:listOfURIs	0..1	
description	xs:string	0..1	
mgmtSchema	xs:anyURI	0..1	

NOTE: objectAttribute is defined in the specializations of mgmtObj. See Annex D.

6.3.5.35 m2m:announcedMgmtResource

This data type is used as base type of the announced variants of announceable <*mgmtObj*> specializations specified in Annex D. It includes the attributes commonly used by all announced <*mgmtObj*> resource type specializations. Note that some specializations of the <*mgmtObj*> resource are announceable while others are not announceable.

It consists of the common attributes included in m2m:announcedResource as defined Table 6.3.6-2 and the additional common attributes shared by all announced <*mgmtObj*> specializations shown in Table 6.3.5.35-1.

Table 6.3.5.35-1: Type Definition of m2m:announcedMgmtResource

Element Path	Element Data Type	Multiplicity	Note
@resourceName	m2m:resourceName	1	common attributes as defined in m2m:announcedResource, see Table 6.3.6-2
resourceType	m2m:resourceType	1	
resourceID	m2m:ID	1	
parentID	m2m:nhURI	1	
creationTime	m2m:timestamp	1	
lastModifiedTime	m2m:timestamp	1	
labels	m2m:labels	0..1	
accessControlPolicyIDs	m2m:acpType	1	
expirationTime	m2m:timestamp	1	
link	xs:anyURI	1	
dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
mgmtDefinition	m2m:mgmtDefinition	1	
objectIDs	m2m:listOfURIs	0..1	
objectPaths	m2m:listOfURIs	0..1	
description	xs:string	0..1	
mgmtSchema	xs:anyURI	0..1	

NOTE: objectAttribute is defined in the specializations of mgmtObj. See Annex D.

6.3.5.36 m2m:contentRef

A complex type that represents the content reference with an associated reference name and URI of the referenced resource.

Table 6.3.5.36-1: Type Definition of m2m:contentRef

Element Path	Element Data Type	Multiplicity	Note
URIReference	(anonymous)	1..n	
URIReference/name	xs:NCName	1	When the URIReference element occurs with multiplicity > 1, the value of each URIReference/name element shall be unique
URIReference/URI	xs:anyURI	1	hierarchical or non-hierarchical resourceID of a <contentInstance> resource

6.3.5.37 m2m:deletionContexts

Table 6.3.5.37-1: Type Definition of m2m:deletionContexts

Element Path	Element Data Type	Multiplicity	Note
timeOfDay	m2m:scheduleEntry	0..n	
locationRegions	m2m:locationRegion	0..n	

6.3.5.38 m2m:flexContainerResource

This data type is used as base type of all <flexContainer> specializations listed in clause 9.6.1.2.2 of oneM2M TS-0001 [6]. It includes the attributes commonly used by all <flexContainer> resource type specializations.

It consists of the common attributes shown in Table 7.4.37.1-1 and the resource specific attributes shared by all <*flexContainer*> specializations shown in Table 7.4.37.1-2.

Table 6.3.5.38-1: Type Definition of m2m:flexContainerResource

Element Path	Element Data Type	Multiplicity	Note
@resourceName	m2m:resourceName	1	
resourceType	m2m:resourceType	1	
resourceID	m2m:ID	1	
parentID	m2m:nhURI	1	
creationTime	m2m:timestamp	0..1	
lastModifiedTime	m2m:timestamp	0..1	
labels	m2m:labels	0..1	
accessControlPolicyIDs	m2m:acpType	0..1	
expirationTime	m2m:timestamp	0..1	
dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
announceTo	list of xs:anyURI	0..1	
announcedAttribute	list of xs:NCName	0..1	
stateTag	xs:nonNegativeInteger	1	
creator	m2m:ID	0..1	
containerDefinition	xs:anyURI	1	
ontologyRef	xs:anyURI	0..1	
contentSize	xs:nonNegativeInteger	1	
nodeLink	xs:anyURI	0..1	

6.3.5.39 m2m:announcedFlexContainerResource

This data type is used as base type of the announced variants of announceable <flexContainer> specializations listed in clause 9.6.1.2.2 of oneM2M TS-0001 [6]. It includes the attributes commonly used by all announced <flexContainer> resource type specializations. Note that some specializations of the <flexContainer> resource are announceable while others are not announceable.

It consists of the common attributes shown in Table 7.4.37.1-1 and the resource specific attributes shared by all announced <*flexContainer*> specializations shown in Table 7.4.37.1-2.

Table 6.3.5.39-1: Type Definition of m2m:announcedFlexContainerResource

Element Path	Element Data Type	Multiplicity	Note
@resourceName	m2m:resourceName	1	
resourceType	m2m:resourceType	1	
resourceID	m2m:ID	1	
parentID	m2m:nhURI	1	
creationTime	m2m:timestamp	0..1	
lastModifiedTime	m2m:timestamp	0..1	
labels	m2m:labels	0..1	
accessControlPolicyIDs	m2m:acpType	1	
expirationTime	m2m:timestamp	0..1	
link	xs:anyURI	1	
dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
stateTag	xs:nonNegativeInteger	0..1	
containerDefinition	xs:anyURI	1	
ontologyRef	xs:anyURI	0..1	
contentSize	xs:nonNegativeInteger	0..1	
nodeLink	xs:anyURI	0..1	

6.3.5.40 m2m:missingData

Used for the *eventNotificationCriteria* attribute of the <subscription> resource. The condition only applies to subscribed-to resources of type <*timeSeries*>. If this condition is set in a <subscription> resource, it enables the AE to keep track of the number of missing data points (i.e. the "number" of the *missingData*) and the corresponding time-stamps over a predefined but renewable duration (i.e. the "duration" of the *missingData*). See clause 7.5.1.2.9.

Table 6.3.5.40-1: Type Definition of m2m:missingData

Element Path	Element Data Type	Multiplicity	Note
number	xs:positiveInteger	1	
duration	xs:duration	1	

6.3.5.41 m2m:tokenPermission

Used in m2m:tokenPermissions.

Table 6.3.5.41-1: Type Definition of m2m:tokenPermission

Element Path	Element Data Type	Multiplicity	Note
resourceIDs	list of m2m:ID	0..1	
privileges	m2m:setOfAcrs	0..1	
roleIDs	list of m2m:roleID	0..1	

6.3.5.42 m2m:tokenClaimSet

This data type defines how to represent the claim set of oneM2M JSON Web Tokens (JWT) required for generating secured tokens (i.e. signed or encrypted representations of JWTs) as defined in clause 7.3.2.6 of oneM2M TS-0003 [7]. Usage of this datatype is specified in oneM2M TS-0003 [7].

Table 6.3.5.42-1: Type Definition of m2m:tokenClaimSet

Element Path	Element Data Type	Multiplicity	Note
version	xs:string	1	
tokenID	m2m:tokenID	1	
holder	m2m:ID	1	
issuer	m2m:ID	1	
notBefore	m2m:timestamp	1	
notAfter	m2m:timestamp	1	
tokenName	xs:string	0..1	
audience	list of m2m:ID	0..1	
permissions	m2m:tokenPermissions	0..1	
extension	xs:string	0..1	

6.3.5.43 m2m:dynAuthLocalTokenIdAssignments

Used for the ***Assigned Token Identifiers*** parameter of a response primitive.

Table 6.3.5.43-1: Type Definition of m2m:dynAuthLocalTokenIdAssignments

Element Path	Element Data Type	Multiplicity	Note
localTokenIdAssignment	(anonymous)	1..n	
localTokenIdAssignment/localTokenID	xs:NCName	1	
localTokenIdAssignment/tokenID	m2m:tokenID	1	

6.3.5.44 m2m:dynAuthTokenSummary

Used for the Token Summary provided by a Dynamic Authorization Server to an Originator.

Table 6.3.5.44-1: Type Definition of m2m:dynAuthTokenSummary

Element Path	Element Data Type	Multiplicity	Note
tokenID	m2m:tokenID	1	
notBefore	m2m:timestamp	1	
notAfter	m2m:timestamp	1	
tokenName	xs:string	0..1	Human readable
audience	list of m2m:ID	0..1	

6.3.5.45 m2m:dynAuthTokenReqInfo

Used for the Token Summary provided by a Dynamic Authorization Server to an Originator.

Table 6.3.5.45-1: Type Definition of m2m:dynAuthTokenReqInfo

Element Path	Element Data Type	Multiplicity	Note
dasInfo	(anonymous)	1..n	
dasInfo/URI	xs:anyURI	1	Dynamic Authorization Server URI
dasInfo/dasRequest	m2m:dynAuthDasRequest	0..1	Information to send to the DAS
dasInfo/securedDasRequest	m2m:dynAuthJWT	0..1	Secured Information to send to the DAS

6.3.5.46 m2m:dynAuthDasRequest

Used in m2m:securityInfo.

Table 6.3.5.46-1: Type Definition of m2m:dynAuthDasRequest

Element Path	Element Data Type	Multiplicity	Note
originator	m2m:ID	1	
targetedResourceType	m2m:resourceType	1	
operation	m2m:operation	1	
originatorIP	(anonymous)	0..1	
originatorIP/ipv4Address	m2m:ipv4	0..1	These elements shall only be present if the originatorIP parent element is present. Exactly one of these elements shall be present
originatorIP/ipv6Address	m2m:ipv6	0..1	
originatorLocation	m2m:locationRegion	0..1	
originatorRoleIDs	list of m2m:roleID	0..1	
requestTimestamp	m2m:absRelTimestamp	0..1	
targetedResourceID	xs:anyURI	0..1	
proposedPrivilegesLifetime	m2m:absRelTimestamp	0..1	
roleIDsFromACPs	list of m2m:roleID	0..1	
tokenIDs	list of m2m:tokenID	0..1	
authorSignIndicator	xs:boolean	0..1	

6.3.5.47 m2m:dynAuthDasResponse

Used in m2m:securityInfo.

Table 6.3.5.47-1: Type Definition of m2m:dynAuthDasResponse

Element Path	Element Data Type	Multiplicity	Note
dynamicACPInfo	(anonymous)	0..1	
dynamicACPInfo/ grantedPrivileges	m2m:setOfAcrs	0..1	These elements shall only be present if the dynamicACPInfo parent element is present. Otherwise they shall not.
dynamicACPInfo/ privilegesLifetime	m2m:absRelTimestamp	0..1	
tokens	list of m2m:dynAuthJWT	0..1	
authorSignReqInfo	xs:boolean	0..1	

6.3.5.48 m2m:securityInfo

Used for the Global Element m2m:securityInfo included into the ***Content*** primitive parameter.

Table 6.3.5.48-1: Type Definition of m2m:securityInfo

Element Path	Element Data Type	Multiplicity	Note
securityInfoType	m2m:securityInfoType	1	
dasRequest	m2m:dynAuthDasRequest	0..1	This element shall only be present if securityInfoType is set to a value of "1" (Dynamic Authorization Request)
dasResponse	m2m:dynAuthDasResponse	0..1	This element shall only be present if securityInfoType is set to a value of "2" (Dynamic Authorization Response)
esprimRandObject	m2m:receiverESPrimRandObject	0..1	This element shall only be present if securityInfoType is set to a value of "4" (receiverESPrimRandObject Response)
esprimObject	m2m:e2eCompactJWE	0..1	This element shall only be present if securityInfoType is set to a value of "5" (ESPrim Object)
escertkeMessage	xs:base64Binary	0..1	This element shall only be present if securityInfoType is set to a value of "6" (ESCertKE Message)
dynAuthRelMapRequest	m2m:dynAuthRelMapRequest	0..1	This element shall only be present if securityInfoType is set to a value of "7" (Dynamic Authorization Relationship Mapping Request)
dynAuthRelMapResponse	m2m:dynAuthRelMapResponse	0..1	This element shall only be present if securityInfoType is set to a value of "8" (Dynamic Authorization Relationship Mapping Response)
NOTE: If securityInfoType is set to a value of "3" (receiverESPrimRandObject Request), no other optional element is included into the global element m2m:securityInfo.			

6.3.5.49 m2m:listOfChildResourceRef

Table 6.3.5.49-1: Type Definition of m2m:listOfChildResourceRef

Element Path	Element Data Type	Multiplicity	Note
resourceRef	m2m:childResourceRef	1..n	References of resources with name and type

6.3.5.50 m2m:originatorESPrimRandObject

Used for the originatorESPrimRandObject parameter in End-to-End Security of Primitives.

Table 6.3.5.50-1: Type Definition of m2m:originatorESPrimRandObject

Element Path	Element Data Type	Multiplicity	Note
esprimRandID	xs:NCName	1	
esprimRandValue	xs:NCName	1	
esprimRandExpiry	m2m:absRelTimestamp	1	See clause 6.3.3
esprimKeyGenAlgID	m2m:esprimKeyGenAlgID	1	See clause 6.3.4.2.40
esprimProtocolAndAlgIDs	list of m2m:esprimProtocolAndAlgID	1	See clause 6.3.4.2.41

6.3.5.51 m2m:receiverESPrimRandObject

Used in m2m:securityInfo.

Table 6.3.5.51-1: Type Definition of m2m:receiverESPrimRandObject

Element Path	Element Data Type	Multiplicity	Note
esprimRandID	xs:NCName	1	
esprimRandValue	xs:NCName	1	
esprimRandExpiry	m2m:absRelTimestamp	1	See clause 6.3.3
esprimKeyGenAlgIDs	list of m2m:esprimKeyGenAlgID	1	See clause 6.3.4.2.40
esprimProtocolAndAlgIDs	list of m2m:esprimProtocolAndAlgID	1	See clause 6.3.4.2.41

6.3.5.52 m2m:e2eSecInfo

Used for the *e2eSecInfo* attribute of the <CSEBase>, <AE> and <remoteCSE> resources.

Table 6.3.5.52-1: Type Definition of m2m:e2eSecInfo

Element Path	Element Data Type	Multiplicity	Note
supportedE2ESecFeatures	list of m2m:suid	1	See clause 6.3.4.2.39
certificates	list of xs:base64Binary	0..1	
sharedReceiverESPrimRandObject	m2m:receiverESPrimRandObject	0..1	See clause 6.3.5.51

6.3.5.53 m2m:tokenPermissions

Used for the *permissions* attribute of the <token> resource and for the *permissions* element in m2m:tokenClaimSet.

Table 6.3.5.53-1: Type Definition of m2m:tokenPermissions

Element Path	Element Data Type	Multiplicity	Note
permission	m2m:tokenPermission	0..n	

6.3.5.54 m2m:backOffParameters

Used for the *backOffParameters* attribute of the <cmdhNwAccessRule> resource.

Table 6.3.5.54-1: Type Definition of m2m:backOffParameters

Element Path	Element Data Type	Multiplicity	Note
backOffParametersSet	(anonymous)	1..n	
backOffParametersSet/networkAction	m2m:networkAction	0..1	
backOffParametersSet/initialBackoffTime	xs:nonNegativeInteger	1	Unit: ms
backOffParametersSet/additionalBackoffTime	xs:nonNegativeInteger	1	Unit: ms
backOffParametersSet/maximumBackoffTime	xs:nonNegativeInteger	1	Unit: ms
backOffParametersSet/optionalRandomBackoffTime	xs:nonNegativeInteger	0..1	Unit: ms

6.3.5.55 m2m:listOfDataLinks

Table 6.3.5.55-1: Type Definition of m2m:listOfDataLinks

Element Path	Element Data Type	Multiplicity	Note
dataLinkEntry	m2m:dataLink	1..n	This data type is a list of triples, each triple containing the following fields (see note): 1) A text string with the name of a data link 2) A URI of the resource (container or flexContainer) that holds the data 3) A text field for identifying simple-type data
NOTE: For usage of this datatype see clauses J.2 and J.3.			

6.3.5.56 m2m:dataLink

Table 6.3.5.56-1: Type Definition of m2m:dataLink

Element Path	Element Data Type	Multiplicity	Note
name	xs:anyURI	1	The name attribute represents the name of the input- / outputDataPoint or operationInput / -Output. It contains the URI of the class name as specified in the ontology (see note)
dataContainerID	m2m:ID	1	URI of the referenced <container> or <flexContainer> resource instance
attributeName	xs:NCName	0..1	Contains: (a) the name of a customAttribute in case the URI points to a <flexContainer> or (b) the string "#latest" in case the URI points to a <container>
NOTE: For usage of this datatype see clause 6.3.5.55 and clauses J.2 and J.3.			

6.3.5.57 m2m:operationMonitor

Table 6.3.5.57-1: Type Definition of m2m:operationMonitor

Element Path	Element Data Type	Multiplicity	Note
operations	m2m:accessControlOperations	0..1	See clause 6.3.4.2.29
originator	m2m:ID	0..1	See clause 6.3.3
NOTE: At least one of the above elements (operations and originators) shall be present. If operations is absent, operationMonitor matches any operation. If originator is absent, operationMonitor matches any originator.			

6.3.5.58 m2m:dynAuthRelMapRequest

Used in m2m:securityInfo.

Table 6.3.5.58-1: Type Definition of m2m:dynAuthRelMapRequest

Element Path	Element Data Type	Multiplicity	Note
originator	m2m:ID	0..1	Including the AE-ID of the AE who owns the Tokens.
tokenIDs	list of m2m:tokenID	0..1	
tokens	list of m2m:dynAuthJWT	0..1	
authorSigns	m2m:signatureList	0..1	
authorSignReqInfo	xs:boolean	0..1	
NOTE: At least one of the Token representation elements (*tokenIDs* and *tokens*) shall be present.			

6.3.5.59 m2m:dynAuthRelMapResponse

Used in m2m:securityInfo.

Table 6.3.5.59-1: Type Definition of m2m:dynAuthRelMapResponse

Element Path	Element Data Type	Multiplicity	Note
tokenIDs	list of m2m:tokenID	0..1	
tokens	list of m2m:dynAuthJWT	0..1	
authorSignReqInfo	xs:boolean	0..1	
signature	m2m:signatureList	0..1	Stored in the authorization relationship mapping record.
NOTE: At least one of the Token representation elements (*tokenIDs* and *tokens*) shall be present.			

6.3.5.60 m2m:ipAddress

Used for the *originatorIP* attribute of the <authorizationDecision> resource.

Table 6.3.5.60-1: Type Definition of m2m:ipAddress

Element Path	Element Data Type	Multiplicity	Note
ipv4Address	m2m:ipv4	0..1	IPv4 address
ipv6Address	m2m:ipv6	0..1	IPv6 address

6.3.5.61 m2m:setOfPermissions

Used for the *policies* attribute of the <authorizationPolicy> resource.

Table 6.3.5.61-1: Type Definition of m2m:setOfPermissions

Element Path	Element Data Type	Mutiplicity	Note
privileges	m2m:setOfAcrs	0..n	

6.3.5.62 m2m:representation

Used for the representation element in the notificationEvent element of a notification. Table 6.3.5.62-1 defines what shall be included in the representation element depending on the value of the *notificationContentType* of the <subscription> resource which triggered the notification.

Table 6.3.5.62-1: Elements used for representation element

Value of notificationContentType	Name of Global Element	Defined in
1, 2	m2m:<resourceType> {other namespace identifier}:<resourceType>	CDT-<resourceType>-v3_11_0.xsd
3	m2m:URI	CDT-responsePrimitive-v3_11_0.xsd
4	m2m:triggerPayload	CDT-triggerPayload-v3_11_0.xsd

The XML representation element shall include a root element which is associated with an XSD Global Element. The root element shall be prefixed with a namespace prefix identifier (e.g. *m2m:*) specified in the associated XSD which defines the respective Global Element. The representation element allows the inclusion of namespaces other than m2m.

6.3.5.63 m2m:sessionDescriptions

Used for the *offeredSessionDescriptions* and *acceptedSessionDescriptions* attributes in the <multimediaSession> resource.

Table 6.3.5.63-1: Type Definition of m2m:sessionDescriptions

Element Path	Element Data Type	Mutiplicity	Note
sessionDescription	m2m:sessionDescription	1..n	

6.3.5.64 m2m:activityPatternElements

Used for the ***activityPatternElements*** attribute of the <AE> and <remoteCSE> resources.

Table 6.3.5.64-1: Type Definition of m2m:activityPatternElements

Element Path	Element Data Type	Mutiplicity	Note
activityPatternElements	m2m:activityPattern	0..n	

6.3.5.65 m2m:activityPattern

Used for describing the anticipated availability of an AE or CSE for communications such as timing pattern, mobility status and expected data size.

Table 6.3.5.65-1: Type Definition of m2m:activityPattern

Element Path	Element Data Type	Mutiplicity	Note
scheduleElement	m2m:scheduleEntries	1..n	
stationaryIndication	m2m:stationaryIndication	0..1	No Default
dataSizeIndicator	xs:positiveInteger	0..1	Unit:Byte

6.3.5.66 m2m:eventNotificationCriteriaSet

Used for *eventNotificationCriteriaSet* attribute in <crossResourceSubscription> resource.

Table 6.3.5.66-1: Type Definition of m2m:eventNotificationCriteriaSet

Element Path	Element Data Type	Mutiplicity	Note
eventNotificationCriteriaEntry	m2m:eventNotificationCriteria	1..n	This data type is a list of eventNotificationCriteria, which is defined in clause 6.3.5.7. Each eventNotificationCriteriaEntry is applied, in order, to a regular resource entry contained in the *regularResourcesAsTarget* attribute of <crossResourceSubscription> resource.

6.3.5.67 m2m:specializationType

Table 6.3.5.67-1: Type Definition of m2m: specializationType

Element Path	Element Data Type	Multiplicity	Note
containerDefinition	xs:anyURI	0..1	
mgmtDefinition	m2m:mgmtDefinition	0..1	

This is an xs:choice. A specializationType shall contain either:

1) a *containerDefinition* element, in which case *mgmtDefinition* shall not appear, or

2) a *mgmtDefinition* element, in which case *containerDefinition* shall not appear.

6.3.5.68 m2m:mashupMembers

Used for the *mashupMember* attribute of the <semanticMashupInstance> resource.

Table 6.3.5.68-1: Type Definition of m2m:mashupMembers

Element Path	Element Data Type	Multiplicity	Note
mashupMember	(anonymous)	1..n	
mashupMember/memberURI	xs:anyURI	1	
mashupMember/memberValue	xs:double	0..1	

6.3.6 Universal and Common attributes

oneM2M TS-0001 [6] defines a number of Universal Attributes (which appear in all resources) and Common Attributes (which appear in more than one resource and have the same meaning whenever they do appear). Their types and values are described in Table 6.3.6-1.

If a Resource is represented as an XML document then the resource attributes (if present) appear in the order listed in this table. They appear before any resource-specific attributes.

Table 6.3.6-1: Universal and Common Attributes

Attribute Name	Data Type	Value restrictions and Notes
resourceType	m2m:resourceType	This attribute is only determined at creation time by the Hosting CSE
resourceID	m2m:ID	This attribute is determined at creation time by the Hosting CSE and used for the non-hierarchical addressing method
parentID	m2m:nhURI	This attribute is determined by the Hosting CSE and specified in all resource types. For <CSEBase> however, the value of this attribute shall be an empty string
creationTime	m2m:timestamp	This attribute is determined by the Hosting CSE when the resource is created locally
lastModifiedTime	m2m:timestamp	This attribute is determined by the Hosting CSE when the addressed resource is modified by means of the UPDATE operation
labels	m2m:labels	Absence of this attribute means there are no labels
accessControlPolicyIDs	m2m:acpType	accessControlPolicyIDs
expirationTime	m2m:timestamp	expirationTime
link	xs:anyURI	Absence of this attribute means that this is not an announced resource
announceTo	list of xs:anyURI	Absence of this attribute means that this is not an announced resource
announcedAttribute	list of xs:NCName	Absence of this attribute means that this is not an announced resource
stateTag	xs:nonNegativeInteger	This attribute is determined by the Hosting CSE. When a resource is created this counter is set to 0 and it will be incremented on every modification of the resource
resourceName	m2m:resourceName	
dynamicAuthorizationConsultationIDs	list of xs:anyURI	dynamicAuthorizationConsultationIDs
creator	m2m:ID	The AE-ID or CSE-ID of the entity which created the resource containing this attribute

Table 6.3.6-2 describes some complex types that group together the universal and common attributes, to be used by Resource Type definitions. Note that *stateTag* and *creator* only appear in a limited number of resource types, and therefore are not included in these definitions, instead they are declared in the corresponding XSD files of the resources that support them.

Table 6.3.6-2: Complex Data Types declaring groups of resource common attributes

XSD type name	Child Elements	Child Element Datatype	Multiplicity	Description
m2m:resource	@resourceName	m2m:resourceName	1	
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
m2m:regularResource	@resourceName	m2m:resourceName	1	Declares the universal / common attributes included in the non-announceable resource types.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	accessControlPolicyIDs	m2m:acpType	0..1	
	expirationTime	m2m:timestamp	1	
	dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
m2m:announceableResource	@resourceName	m2m:resourceName	1	Declares the universal / common
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	

XSD type name	Child Elements	Child Element Datatype	Multiplicity	Description
	parentID	xs:anyURI	1	attributes included in the majority of announceable resource types.
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	accessControlPolicyIDs	m2m:acpType	0..1	
	expirationTime	m2m:timestamp	1	
	dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
	announceTo	list of xs:anyURI	0..1	
	announcedAttribute	list of xs:NCName	0..1	
m2m:announcedResource	@resourceName	m2m:resourceName	1	Declares the universal / common attributes in the announced variant of the preceding resources.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	accessControlPolicyIDs	m2m:acpType	1	
	expirationTime	m2m:timestamp	1	
	link	xs:anyURI	1	
	dynamicAuthorizationConsultationIDs	list of xs:anyURI	0..1	
m2m:subordinateResource	@resourceName	m2m:resourceName	1	Declares the universal / common attributes included in the non-announceable resource types without accessControlPolicyIDs.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	expirationTime	m2m:timestamp	1	
m2m:announceableSubordinateResource	@resourceName	m2m:resourceName	1	Declares the universal / common attributes used by resource types that are subordinate children of other resources.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	xs:anyURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	expirationTime	m2m:timestamp	1	
	announceTo	list of xs:anyURI	0..1	
	announcedAttribute	list of xs: NCName	0..1	
m2m:announcedSubordinateResource	@resourceName	m2m:resourceName	1	Declares the common / universal attributes used in the announced variants of the subordinate resource types.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labels	0..1	
	expirationTime	m2m:timestamp	1	
	link	xs:anyURI	1	

NOTE: In Table 6.3.6-2, names of XML schema attributes are prefixed with a "@" character to differentiate these from Resource attribute names. The "@" character is not part of the actual attribute name.

6.4 Message parameter data types

6.4.1 Request primitive parameter data types

The data types of request primitive parameters are specified in this clause.

Detailed request primitive parameter descriptions and usage can be found in clause 8.1.2 of the oneM2M TS-0001 [6]. Further details on the representation of request primitives are specified in clauses 7.2.1.1 and 8 of the present document.

Table 6.4.1-1 shows the structure of the request primitive. This is defined as the m2m:requestPrimitive element in the XSD file CDT-requestPrimitive-v3_11_0.xsd.

Table 6.4.1-1: Data Types for Request primitive parameters

Primitive Parameter	Data Type	Multiplicity	Default Handling (note 2)	Note
Operation	m2m:operation	1	Not applicable	See clause 6.3.4.2.5
To	xs:anyURI	1	Not applicable	
From	m2m:ID	0..1	Not applicable	See clause 6.3.3 Also see note 2 below
Request Identifier	m2m:requestID	1	Not applicable	See clause 6.3.3
Resource Type	m2m:resourceType	0..1	No default	See clause 6.3.4.2.1
Content	m2m:primitiveContent	0..1	No default	See clause 6.3.5.5
Role IDs	list of m2m:roleID	0..1	Not applicable	
Originating Timestamp	m2m:timestamp	0..1	No default	
Request Expiration Timestamp	m2m:absRelTimestamp	0..1	Can be given by CMDH policy (clause D.12)	"Result Expiration Timestamp" shall be later than "Request Message Expiration Timestamp"
Result Expiration Timestamp	m2m:absRelTimestamp	0..1	Can be given by CMDH policy (clause D.12)	
Operation Execution Time	m2m:absRelTimestamp	0..1	Can be given by CMDH policy (clause D.12)	
Response Type	m2m:responseTypeInfo	0..1	Use "blockingRequest"	See clause 6.3.5.30
Result Persistence	m2m:absRelTimestamp	0..1	Can be given by CMDH policy (clause D.12)	
Result Content	m2m:resultContent	0..1	The default value depends on a given operation. See Table 8.1.2-1 of oneM2M TS-0001 [6]	See clause 6.3.4.2.7
Event Category	m2m:eventCat	0..1	No default	See clause 6.3.3
Delivery Aggregation	xs:boolean	0..1	Can be given by CMDH policy (clause D.12), otherwise false	
Group Request Identifier	xs:string	0..1	No default	
Filter Criteria	m2m:filterCriteria	0..1	No default	See clause 6.3.5.8
Desired Identifier Result Type	m2m:desIdResType	0..1	Use "structured"	See clause 6.3.4.2.8
Tokens	list of m2m:dynAuthJWT	0..1	Not applicable	See clause 6.3.3
Token IDs	list of m2m:tokenID	0..1	Not applicable	
Local Token IDs	list of xs:NCName	0..1	No default	
Token Request Indicator	xs:boolean	0..1	No default	
Group Request Target Members	list of xs:anyURI	0..1	No default	
Authorization Signature Indicator	xs:boolean	0..1	No default	
Authorization Signatures	m2m:signatureList	0..1	No default	See clause 6.3.3
Authorization Relationship Indicator	xs:boolean	0..1	No default	
Semantic Query Indicator	xs:boolean	0..1	No default	Semantic Query Indicator

Primitive Parameter	Data Type	Multiplicity	Default Handling (note 2)	Note
Release Version Indicator	m2m:releaseVersion	1	No default	This parameter is set to the release version that the primitive complies with
Vendor Information	xs:string	0..1	No default	This parameter is used to convey vendor specific information. No procedures are defined
NOTE 1: Default handling is the request handling procedure on a Transit/Hosting CSE when the request parameter is not included in a request primitive. This is not applicable for mandatory parameters which are marked as 'M' in Table 7.2.1.1-1. NOTE 2: The ***From*** parameter shall be present for all requests except for <AE> CREATE where it is optional.				

6.4.2 Response primitive parameter data types

The data types of response primitive parameters are specified in this clause.

Detailed response message parameter descriptions and usage can be found in clause 8.1.3 of oneM2M TS-0001 [6]. Further details on the representation of response primitives can be found in clauses 7.2.1.2 and 8 of the present document.

Table 6.4.2-1 shows the structure of the response primitive. This is defined as the m2m:responsePrimitive element in the XSD file CDT-responsePrimitive-v3_11_0.xsd.

Table 6.4.2-1: Data Types for Response primitive parameters

Primitive Parameter	Data Type	Multiplicity	Note
Response Status Code	m2m:responseStatusCode	1	See clause 6.3.4.2.9
Request Identifier	m2m:requestID	1	See clause 6.3.3
Content	m2m:primitiveContent	0..1	See clause 6.3.5.5
To	m2m:ID	0..1	See clause 6.3.3
From	m2m:ID	0..1	
Originating Timestamp	m2m:timestamp	0..1	See Table 6.3.3-1
Result Expiration Timestamp	m2m:absRelTimestamp	0..1	See Table 6.3.3-1
Event Category	m2m:eventCat	0..1	See clause 6.3.3
Content Status	m2m:contentStatus	0..1	See clause 6.3.4.2.44
Content Offset	xs:positiveInteger	0..1	
Assigned Token Identifiers	m2m:dynAuthLocalTokenIdAssignments	0..1	See clause 6.3.5.43
Token Request Information	m2m:dynAuthTokenReqInfo	0..1	See clause 6.3.5.45
Authorization Signature Request Information	xs:boolean	0..1	
Release Version Indicator	m2m:releaseVersion	1	This parameter is not present when a response is targeting a Release-1 entity and shall be included in all other cases
Vendor Information	xs:string	0..1	This parameter is used to convey vendor specific information. No procedures are defined

6.5 Resource data types

6.5.1 Description

Each oneM2M Resource Data Type is defined using XML Schema (XSD), and supplied as a separate XSD document. This XML Schema defines the attributes of the Resource in accordance with oneM2M TS-0001 [6]. It represents an entire resource. In other words if and only if a requestor retrieves an entire resource in XML format, the XML that is returned shall be valid with respect to the schema for that resource. Note that the payload of a Create or Update request primitive does not necessarily have to be valid according to the schema, as this payload is not required to contain values for all the resource attributes. For example, a resource might contain mandatory read-only attributes which do not appear in a Create or Update request.

Each Resource Type, along with its Announced variant (if there is one) is defined in a separate XSD file. The name of that file should be prefixed with "CDT-" and followed by the resource type name and version of the present document.

The individual Resource Types inherit from a set of base resource types. These definitions, which can be found in the file CDT-commonTypes-v3_11_0.xsd, contain definitions for the common and universal attributes, and establish an inheritance hierarchy shown in Figure 6.5.1-1.

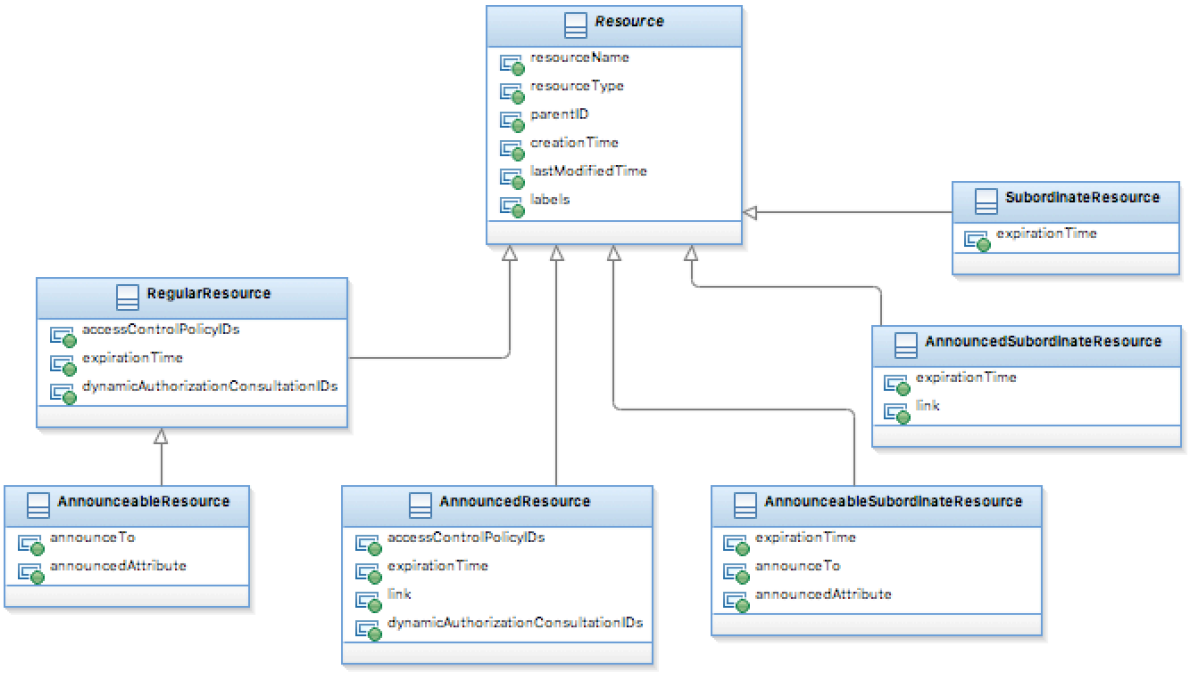

Figure 6.5.1-1: Resource Types

6.5.2 resource

6.5.2.1 Description

This XSD type definition includes the six universal attributes that are present in all oneM2M resource type definitions. It forms the root of the resource inheritance hierarchy.

6.5.2.2 Reference

See Table 6.3.6-2.

6.5.2.3 Usage

This type is used indirectly by all resource types. It is used directly only by the <CSEBase> resource type.

6.5.3 regularResource

6.5.3.1 Description

This type definition includes the universal and common attributes used by the non-announceable oneM2M resources.

6.5.3.2 Reference

See Table 6.3.6-2.

6.5.3.3 Usage

This type is used by the following resource types:

<delivery>, <eventConfig>, <execInstance>, <m2mServiceSubscriptionProfile>, <mgmtCommand>, <request>, <serviceSubscribedNode>, <statsCollect>, <statsConfig>, <subscription>, <serviceSubscribedAppRule>, <notificationTargetMgmtPolicyRef>, <notificationTargetPolicy>, <policyDeletionRules>, <dynamicAuthorizationConsultation>, <role>, <token>, <authorizationDecision>, <authorizationPolicy> <authorizationInformation>, <AEContactList>, <AEContactListPerCSE>, <localMulticastGroup>, <triggerRequest>, <crossResourceSubscription>, <backgroundDataTransfer>, <transactionMgmt>, <transaction>.

6.5.4 announceableResource

6.5.4.1 Description

This type definition includes the universal and common attributes used by oneM2M resource types that are capable of being announced. In addition to the attributes of a regularResource, it includes (as optional) the common attributes that are used by the announcement mechanism.

6.5.4.2 Reference

See Table 6.3.6-2.

6.5.4.3 Usage

This type is used by the following resource types:

<AE>, <container>, <group>, <locationPolicy>, <node>, <remoteCSE>, <semanticDescriptor>, <timeSeries>, <ontologyRepository>, <ontology>, <semanticMashupJobProfile>, <semanticMashupInstance>, <semanticMashupResult>, <multimediaSession>, <schedule>.

It is also used by the specializations of <mgmtObj>.

6.5.5 announcedResource

6.5.5.1 Description

This type definition includes the universal and common attributes used by a resource that is announcing an announceable resource. In addition to the attributes of a regularResource, it includes (as optional) the *link* common attribute.

6.5.5.2 Reference

See Table 6.3.6-2.

6.5.5.3 Usage

This type is used by the following resource types:

<AEAnnc>, <containerAnnc>, <groupAnnc>, <locationPolicyAnnc>, <nodeAnnc>, <remoteCSEAnnc>, <semanticDescriptorAnnc>, <[flexContainer]Annc>, <timeSeriesAnnc>, <ontologyRepositoryAnnc>, <ontologyAnnc>, <semanticMashupJobProfileAnnc>, <semanticMashupInstanceAnnc>, <semanticMashupResultAnne>, <multimediaSessionAnnc>,<scheduleAnnc>.

It is also used by the xxxAnnc variants of the <mgmtObj> specializations.

6.5.6 announceableSubordinateResource

6.5.6.1 Description

This type definition includes the common attributes used by resource types that are subordinate children of other resource types. It excludes attributes like *accessControlPolicyIDs*, as this attribute is defined for such resources.

6.5.6.2 Reference

See Table 6.3.6-2.

6.5.6.3 Usage

This type is used by the following resource types:

<accessControlPolicy>, <contentInstance>, <timeSeriesInstance>.

It is also used by the xxxAnnc variants of the <mgmtObj> specializations.

6.5.7 announcedSubordinateResource

6.5.7.1 Description

This type definition includes the common attributes used by the Announced variants of the resource types that are subordinate children of other resource types.

6.5.7.2 Reference

See Table 6.3.6-2.

6.5.7.3 Usage

This type is used by the following resource types:

<accessControlPolicyAnnc>, <contentInstanceAnnc>, <timeSeriesInstanceAnnc>.

6.5.8 subordinateResource

6.5.8.1 Description

This type definition includes the universal and common attributes used by the non-announceable M2M resources except the *accessControlPolicyIDs* attribute.

6.5.8.2 Reference

See Table 6.3.6-2.

6.5.8.3 Usage

This type is used by the following resource types:

<pollingChannel>.

6.6 Response status codes

6.6.1 Introduction

The present clause specifies the assignment of oneM2M Response Status Code (RSC) values, which are returned in the Response Status Code parameter of Response primitive.

The RSC may be delivered as oneM2M defined structured data, or the mapped native status code for transport protocol binding (e.g. HTTP, CoAP, MQTT).

6.6.2 RSC framework overview

The RSCs are categorized as one of 6 classes:

Table 6.6.2-1: Definition of Response Status Code class

Status Class	Code Class	Interpretation
Informational	1xxx	The request is successfully received, but the request is still in progress.
Success	2xxx	The request is successfully received, understood, and accepted.
Redirection	3xxx	(Not used in present release)
Originator Error	4xxx	The request was malformed by the Originator and, is rejected.
Receiver Error	5xxx	The requested operation cannot be performed due to an error condition at the Receiver CSE.
Network Service Error	6xxx	The requested operation cannot be performed due to an error condition at the Network Service Entity.

6.6.3 Definition of Response Status Codes

6.6.3.1 Overview

The tables in the following clauses specify the RSCs for oneM2M releases. Each RSC includes: a response status represented as numeric code. The supplemental information may be returned when it is needed.

6.6.3.2 Informational response class

Table 6.6.3.2-1 specifies the RSCs for acknowledgement responses for each release.

Table 6.6.3.2-1: Informational response class

Numeric Code	Description
1000	ACCEPTED
1001	ACCEPTED for nonBlockingRequestSynch
1002	ACCEPTED for nonBlockingRequestAsynch

6.6.3.3 Successful response class

Table 6.6.3.3-1 specifies the RSCs for successful responses.

Table 6.6.3.3-1: RSCs for successful response class

Numeric Code	Description
2000	OK
2001	CREATED
2002	DELETED
2004	UPDATED

6.6.3.4 Redirection response class

In the present document, no values in this response class are defined.

Table 6.6.3.4-1: RSCs for redirection response class

Numeric Code	Description

6.6.3.5 Originator error response class

Table 6.6.3.5-1 specifies the RSCs for Originator error responses.

41xx codes are oneM2M specific.

Table 6.6.3.5-1: RSCs for Originator error response class

Numeric Code	Description
4000	BAD_REQUEST
4001	RELEASE_VERSION_NOT_SUPPORTED
4004	NOT_FOUND
4005	OPERATION_NOT_ALLOWED
4008	REQUEST_TIMEOUT
4015	UNSUPPORTED_MEDIA_TYPE
4101	SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE
4102	CONTENTS_UNACCEPTABLE
4103	ORIGINATOR_HAS_NO_PRIVILEGE
4104	GROUP_REQUEST_IDENTIFIER_EXISTS
4105	CONFLICT
4106	ORIGINATOR_HAS_NOT_REGISTERED
4107	SECURITY_ASSOCIATION_REQUIRED
4108	INVALID_CHILD_RESOURCE_TYPE
4109	NO_MEMBERS
4110	GROUP_MEMBER_TYPE_INCONSISTENT
4111	ESPRIM_UNSUPPORTED_OPTION
4112	ESPRIM_UNKNOWN_KEY_ID
4113	ESPRIM_UNKNOWN_ORIG_RAND_ID
4114	ESPRIM_UNKNOWN_RECV_RAND_ID
4115	ESPRIM_BAD_MAC
4116	ESPRIM_IMPERSONATION_ERROR
4117	ORIGINATOR_HAS_ALREADY_REGISTERED
4118	ONTOLOGY_NOT_AVAILABLE
4119	LINKED_SEMANTICS_NOT_AVAILABLE
4120	INVALID_SEMANTICS
4121	MASHUP_MEMBER_NOT_FOUND
4122	INVALID_TRIGGER_PURPOSE
4123	ILLEGAL_TRANSACTION_STATE_TRANSITION_ATTEMPTED
4124	BLOCKING_SUBSCRIPTION_ALREADY_EXISTS
4125	SPECIALIZATION_SCHEMA_NOT_FOUND

6.6.3.6 Receiver error response class

Table 6.6.3.6-1 specifies the RSCs for Receiver error responses.

51xx codes are oneM2M specific, which are used in generic procedures.

52xx codes are oneM2M specific, which are used in resource specific procedures.

Table 6.6.3.6-1: RSCs for Receiver error response class

Numeric Code	Description
5000	INTERNAL_SERVER_ERROR
5001	NOT_IMPLEMENTED
5103	TARGET_NOT_REACHABLE
5105	RECEIVER_HAS_NO_PRIVILEGE
5106	ALREADY_EXISTS
5203	TARGET_NOT_SUBSCRIBABLE
5204	SUBSCRIPTION_VERIFICATION_INITIATION_FAILED
5205	SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE
5206	NON_BLOCKING_SYNCH_REQUEST_NOT_SUPPORTED
5207	NOT_ACCEPTABLE
5208	DISCOVERY_DENIED_BY_IPE
5209	GROUP_MEMBERS_NOT_RESPONDED
5210	ESPRIM_DECRYPTION_ERROR
5211	ESPRIM_ENCRYPTION_ERROR
5212	SPARQL_UPDATE_ERROR
5214	TARGET_HAS_NO_SESSION_CAPABILITY
5215	SESSION_IS_ONLINE
5216	JOIN_MULTICAST_GROUP_FAILED
5217	LEAVE_MULTICAST_GROUP_FAILED
5218	TRIGGERING_DISABLED_FOR_RECIPIENT
5219	UNABLE_TO_REPLACE_REQUEST
5220	UNABLE_TO_RECALL_REQUEST
5221	CROSS_RESOURCE_OPERATION_FAILURE
5222	TRANSACTION_PROCESSING_IS_INCOMPLETE

6.6.3.7 Network system error response class

Table 6.6.3.7-1 specifies the RSCs for when the external system reported errors over Mcn reference point.

Table 6.6.3.7-1: RSCs for Network system error response class

Numeric Code	Description
6003	EXTERNAL_OBJECT_NOT_REACHABLE
6005	EXTERNAL_OBJECT_NOT_FOUND
6010	MAX_NUMBER_OF_MEMBER_EXCEEDED
6020	MGMT_SESSION_CANNOT_BE_ESTABLISHED
6021	MGMT_SESSION_ESTABLISHMENT_TIMEOUT
6022	INVALID_CMDTYPE
6023	INVALID_ARGUMENTS
6024	INSUFFICIENT_ARGUMENTS
6025	MGMT_CONVERSION_ERROR
6026	MGMT_CANCELLATION_FAILED
6028	ALREADY_COMPLETE
6029	MGMT_COMMAND_NOT_CANCELLABLE
6030	EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_RQET_TIMEOUT
6031	EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_OET_TIMEOUT

6.7 oneM2M specific MIME media types

The present clause defines oneM2M specific MIME media types which may be used by protocol bindings.

The oneM2M specific MIME media types are defined under the vendor tree of the "application" media type which is prefixed with "application/vnd.onem2m-".

Table 6.7-1: oneM2M specific MIME media types

oneM2M specific MIME subtype	mapped oneM2M data type	Note
vnd.onem2m-res+xml	m2m:resource	For oneM2M resource operation. The type of the oneM2M resource in the ***Content*** shall be indicated by the ***Resource Type*** parameter. XML serialization rule is applied. (See clause 7.5.2)
vnd.onem2m-res+json	m2m:resource	Same information of above. JSON serialization rule is applied. (See clause 7.5.2)
vnd.onem2m-res+cbor	m2m:resource	Same information of above. CBOR serialization rule is applied. (See clause 7.5.2)
vnd.onem2m-ntfy+xml	m2m:notification or m2m:aggregatedNotification	For Notify operation for resource subscription. XML serialization rule is applied. (See clause 7.5.1)
vnd.onem2m-ntfy+json	m2m:notification or m2m:aggregatedNotification	Same information of above. JSON serialization rule is applied. (See clause 7.5.1)
vnd.onem2m-ntfy+cbor	m2m:notification or m2m:aggregatedNotification	Same information of above. CBOR serialization rule is applied. (See clause 7.5.1)
vnd.onem2m-preq+xml	m2m:requestPrimitive	For exchanging serialized oneM2M request primitive. XML serialization rule is applied. (See clauses 6.4.1 and 7.2.1.1)
vnd.onem2m-preq+json	m2m:requestPrimitive	Same information of above. JSON serialization rule is applied. (See clauses 6.4.1 and 7.2.1.1)
vnd.onem2m-preq+cbor	m2m:requestPrimitive	Same information of above. CBOR serialization rule is applied. (See clauses 6.4.1 and 7.2.1.1)
vnd.onem2m-prsp+xml	m2m:responsePrimitive	For exchanging Response parameters. XML serialization rules is applied. (See clauses 6.4.2 and 7.2.1.2)
vnd.onem2m-prsp+json	m2m:responsePrimitive	Same information of above. JSON serialization rule is applied. (See clauses 6.4.2 and 7.2.1.2)
vnd.onem2m-prsp+cbor	m2m:responsePrimitive	Same information of above. CBOR serialization rule is applied. (See clauses 6.4.2 and 7.2.1.2)

6.8 Virtual Resources

A virtual resource is used to trigger processing and/or retrieve results, but does not have a permanent representation in a CSE. Table 6.8-1 lists the Virtual Resources.

Table 6.8-1: Virtual Resources

Virtual Resource Type	resourceName	Parent Resource	Notes
<latest>	la	<container>	See clause 7.4.27
<oldest>	ol	<container>	See clause 7.4.28
<fanOutPoint>	fopt	<group>	See clause 7.4.14
<semanticFanOutPoint>	sfop	<group>	See clause 7.4.35
<pollingChannelURI>	pcu	<pollingChannel>	See clause 7.4.22
<notificationTargetSelfReference>	ntsr	<subscription>	See clause 7.4.33
<semanticValidation>	smv	<ontologyRepository>	See clause 7.4.48
<mashup>	msp	<semanticMashupInstance >	See clause 7.4.51

Each resource instance listed in "Parent Resource" column of Table 6.8-1 has one virtual resource child of each type listed against it in the table. These child resource instances have fixed resourceNames, as shown in the second column.

A virtual resource shall be addressed using the hierarchical addressing method formed by taking the structured or unstructured resource identifier of the parent resource and appending a "/" followed by the resourceName of the virtual resource.

7 oneM2M procedures

7.1 Introduction

The following clauses (7.2 to 7.6) describe prerequisites such as primitive format and procedure outlines with three generic scenarios that are Originator, Receiver, and Resource Handling in accordance with CRUD+N operations. In addition, for specific resource types they provide common or resource-specific attributes, data type definitions for the resource-specific attributes, and child resources. They also explain the resource-specific procedures on CRUD+N operations to communicate with oneM2M compliant M2M Platform Systems by oneM2M protocols and APIs as follows:

- Primitive formats and generic procedures
- Common operations
- Resource type-specific definitions and procedures
- Notification definition and procedures

7.2 Primitive format and generic procedure

7.2.1 Primitive format

7.2.1.1 Request primitive format

Table 7.2.1.1-1 summarizes the primitive parameters of the Request primitive, indicating their presence depending on the C, R, U, D or N operations. "M" indicates mandatory, "O" indicates optional, "NP" indicates not present.

Refer to clause 8.1.2 of the oneM2M TS-0001 [6] for additional information on the request primitive parameters.

Table 7.2.1.1-1: Request Primitive Parameters

Primitive Parameter	CREATE	RETRIEVE	UPDATE	DELETE	NOTIFY
Operation	M	M	M	M	M
To	M	M	M	M	M
From	O See note	M	M	M	M
Request Identifier	M	M	M	M	M
Resource Type	M	NP	NP	NP	NP
Content	M	O	M	NP	M
Role IDs	O	O	O	O	O
Originating Timestamp	O	O	O	O	O
Request Expiration Timestamp	O	O	O	O	O
Result Expiration Time	O	O	O	O	O
Operation Execution Time	O	O	O	O	O
Response Type	O	O	O	O	O
Result Persistence	O	O	O	O	NP
Result Content	O	O	O	O	NP
Event Category	O	O	O	O	O
Delivery Aggregation	O	O	O	O	O
Group Request Identifier	O	O	O	O	O
Filter Criteria	NP	O	O	O	NP
Discovery Result Type	NP	O	NP	NP	NP
Tokens	O	O	O	O	O
Token IDs	O	O	O	O	O
Local Token IDs	O	O	O	O	O
Token Request Indicator	O	O	O	O	O
Group Request Target Members	O	O	O	O	NP
Authorization Signature Indicator	O	O	O	O	NP
Authorization Signature	O	O	O	O	NP
Authorization Relationship Indicator	O	O	O	O	NP
Semantic Query Indicator	NP	O	NP	NP	NP
Release Version Indicator	M	M	M	M	M
Vendor Information	O	O	O	O	O
NOTE: The *From* parameter is Mandatory for all requests except for AE CREATE. For AE CREATE, it is Optional.					

The ***Content*** parameter in a Request shall contain one of the following:

1) A partial Resource. This applies to Create and Update request primitives. In the case of Create request the ***Content*** parameter shall contain a single root element whose name is the name of the Resource and whose content consists of one or more attributes, child Resources or childResource references. In the case of an Update request primitive, the ***Content*** parameter shall contain the attribute and new values. Attributes to be deleted from the resource shall be indicated without a value. In both cases the resource type is as defined in clause 7.4, however since a partial resource is being transferred it is not required to be valid according to the XSD for that resource in terms of the presence of resource attributes. Any attribute that is present, however, shall comply to the data type defined in the XSD of that resource.

2) A Notification Data Object. This applies to Notification request primitives. The data type of the data object is named <m2m:notification> and is described in clause 7.5.1.

3) An Aggregated Notification. This applies to Notification request primitives. The data type of the data object is named <m2m:aggregatedNotification> and contains multiple <m2m:notification> objects. This is described in clause 7.5.1.

4) An AttributeList element, as described in clause 7.5.2. This is used in partial retrieve request primitives to indicate a list of attribute names whose values shall be retrieved in the response.

5) A ResponsePrimitive object as described in clause 7.5.1. This applies to Notification request primitives which are sent when accessing resources in asynchronous non-blocking mode.

7.2.1.2 Response primitive format

Table 7.2.1.2-1 summarizes the primitive parameters for Response primitive, indicating their presence depending on the C, R, U, D or N operations of the associated Request primitive and whether this operation was successful or caused an error. "M" indicates mandatory, "O" indicates optional, "NP" indicates not present.

Refer to clause 8.1.3 of oneM2M TS-0001 [6] for additional information on the request primitive parameters.

NOTE: ***Response Code*** and ***Status Code*** parameters are merged into the ***Response Status Code*** parameter.

Table 7.2.1.2-1: Response Primitive Parameters

Primitive parameter	Ack	CREATE Success	RETRIEVE Success	UPDATE Success	DELETE Success	NOTIFY Success	Error
Response Status Code	M	M	M	M	M	M	M
Request Identifier	M	M	M	M	M	M	M
Content	O	O	M	O	O	O	O
To	O	O	O	O	O	O	O
From	O	O	O	O	O	O	O
Originating Timestamp	O	O	O	O	O	O	O
Result Expiration Timestamp	O	O	O	O	O	O	O
Event Category	O	O	O	O	O	O	O
Content Status	NP	NP	O	NP	NP	NP	NP
Content Offset	NP	NP	O	NP	NP	NP	NP
Assigned Token Identifiers	NP	O	O	O	O	O	O
Token Request Information	NP	NP	NP	NP	NP	NP	O
Authorization Signature Request Information	NP	NP	NP	NP	NP	NP	O
Release Version Indicator	M	M	M	M	M	M	M
Vendor Information	O	O	O	O	O	O	O

The Content parameter in a Response shall contain one of the following:

1) A complete or partial Resource. This applies to a response primitive sent in reply to create and retrieve request message. A partial resource also applies to a response primitive sent in reply to update request message. The ***Content*** parameter shall contain a single root element whose name is the name of the Resource and whose content consists of one or more attributes, child resources or childResource references. In this case the resource type is as defined in clause 7.4. However if a partial resource is being transferred, it is not required to be valid according to the XSD for that resource, in terms of the presence of resource attributes. Any attribute that is present, however, shall comply to the data type defined in the XSD of that resource.

2) The URI of a resource. This is included directly as the content of the ***Content*** parameter (like in case 6).

3) A partial resource and its hierarchical URI. These are included in a root element called m2m:resource defined in clause 7.5.2. The URI is included as an attribute of m2m:resource.

4) A list of URIs. This can be used for transferring the childResource URIs in a Discovery response. These are included in an element called m2m:URIList defined in clause 7.5.2.

5) A list of childResourceRef. This can be used for transferring the child resource references in a Discovery response. These are included in an element called m2m:resourceRefList defined in clause 7.5.2.

6) An Aggregated Response. This is sent as a result of a Group operation. This uses the element m2m:aggregatedResponse defined in clause 7.5.2.

7) A request primitive. A pending request is sent in a polling response. This uses the element m2m:requestPrimitive defined in clause 6.4.1.

8) Human-readable error message. This is included in an element called m2m:debugInfo defined in clause 7.5.2.

7.2.2 Description of generic procedures

7.2.2.1 Generic resource request procedure for originator

A generic resource Request procedure shall be comprised of the following actions. Additional actions specific to individual procedures are listed in the respective clauses by referencing these actions and providing additional steps. The Originator shall execute the following steps in order.

Figure 7.2.2.1-1: Generic procedure of Originator

Orig-1.0 "Compose Request primitive": Refer to clause 7.3.1.1 for details.

Orig-2.0 "Send a Request primitive to the Receiver CSE": The Request primitive shall include the mandatory parameters ***Operation***, ***To***, ***From*** and ***Request Identifier***. Refer to clause 7.3.1.2 for details.

Orig-3.0 "Check Response Type": In this step, the Originator checks that the communication method is either blockingRequest, nonBlockingRequestSynch, nonBlockingRequestAsynch or flexBlocking by using the ***Response Type*** parameter (see detail in clause 8.1.2 in the oneM2M TS-0001 [6]). If the ***Response Type*** parameter does not exist, the communication method is "blockingRequest" as specified at clause 6.4.1.

If the ***Response Type*** is blockingRequest the Originator waits for the Response primitive and goes to step Orig-4.0. If the ***Response Type*** is nonBlockingRequestSync, it waits for an acknowledgement Response primitive and goes to step Orig-4.1. If the ***Response Type*** is nonBlockingRequestAsynch, it waits for an acknowledgement Response primitive and goes to step Orig-4.1. If the ***Response Type*** is flexBlocking, the Originator shall wait for a Response primitive as in Orig-4.0 and Orig-4.1 below, if the Response primitive is an acknowledgement it shall proceed according to Orig-4.1 (nonBlockingRequestSynch or nonBlockingRequestAsynch) otherwise it shall proceed according to Orig-4.0 (blockingRequest).

Orig-4.0 and Orig-4.1 "Wait for Response primitive": Refer to clause 7.3.1.3 for details.

Orig-5.0 "Send a Request primitive with op=R": The op=R means Retrieve operation. The Request primitive shall include the mandatory parameters ***Operation***, ***To***, ***From*** and ***Request Identifier***. The ***Response Type*** of the "Request" primitive shall be blockingRequest. See clause 7.3.1.4 for details.

Orig-5.1 "Receive a Response primitive from the Hosting CSE": The Originator shall receive the mandatory parameters which are ***Response Status Code***, ***Request Identifier*** and ***Content*** parameter. The ***Request Identifier*** shall be identical to the value of that parameter from Orig-5.0. The information in the ***Content*** parameter is the result when the Receiver completed handling of the Request primitive of Orig-2.0.

Orig-5.2 "Completion of operation by ***Response Status Code*** parameter": When the ***Response Status Code*** is successful and ***Content*** parameter exists, it goes to Orig-5.3. When the ***Response Status Code*** is acknowledgment which indicates processing at the Receiver, it goes to Orig-5.0. When the ***Response Status Code*** is an error such as Originator error (4XXX) or Receiver error (5XXX) or Network error (6XXX) or the ***Content*** parameter is absent, it goes to finish with error.

Orig-5.3 "Extract a result from Response primitive of Orig-5.1": The information in the *operationResult* attribute of the <request> resource in the ***Content*** parameter from Orig-5.1 is extracted from the Response primitive which included the ***Request Identifier***, ***Response Status Code*** and optional ***Content*** parameters. The <request> resource shall include mandatory attributes as specified in clause 9.6.12 of oneM2M TS-0001 [6]. The ***Request Identifier*** in the *operationResult* attribute shall be identical to that in Orig-2.0.

Orig-6.0 "Process Response primitive": The ***Request Identifier*** shall be identical to that in Orig-2.0. The Originator processes the response.

Orig-7.0 "Receive a Request primitive with op=N": The op=N means Notify operation. The Originator receives a Request primitive with mandatory parameters ***Operation***, ***To***, ***From***, ***Request Identifier*** and ***Content***. The ***Operation*** parameter shall be Notify. The ***Content*** parameter is the notification information as specified in clause 7.5.1.1.

Orig-8.0 "Create a Response primitive": The Originator creates Response primitive with mandatory parameters ***Response Status Code*** and ***Request Identifier***. The ***Request Identifier*** shall be identical to that in Orig-7.0.

Orig-9.0 "Send a Response primitive": The Response primitive which is created at Orig-8.0 shall be sent to the Receiver. Refer to clause 7.3.2.3 for details.

Orig-9.1 "Extract Response primitive of Orig-2.0 from Orig-7.0": The information in the *operationResult* attribute of the <request> resource from Orig-7.0 in Response primitive includes ***Request Identifier***, ***Response Status Code*** and optional ***Content*** parameters. The <request> resource shall include mandatory attributes as specified in clause 9.6.12 of oneM2M TS-0001 [6]. The ***Request Identifier*** in the *operationResult* attribute shall be identical to that in Orig-2.0.

7.2.2.2 Generic procedure for handling a Request at a receiver

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 error response" (refer to clause 7.3.3.13 for details) and then "Send Response primitive" (refer to clause 7.3.2.4 for details). The corresponding ***Response Status Code*** shall be included in the Response primitive.

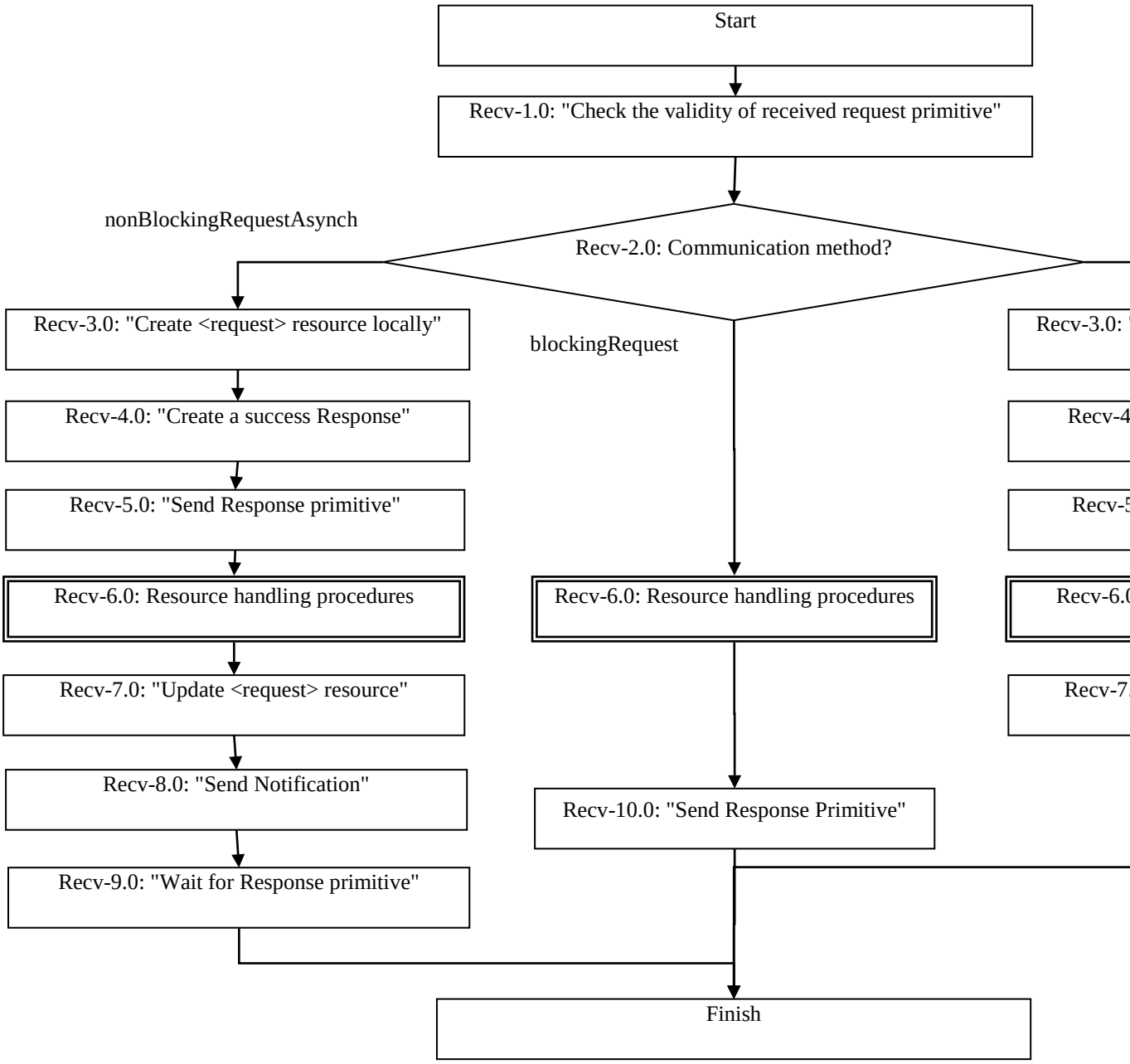

Figure 7.2.2.2-1: Generic procedure of Receiver

Recv-1.0 "Check the validity of received request primitive": See clause 7.3.2.1 for details.

Recv-2.0 "Communication method?": The Receiver CSE checks whether a received request is blockingRequest, nonBlockingRequestSynch or nonBlockingRequestAsynch by using the ***Response Type*** parameter (see detail in clause 8.1.2 in oneM2M TS-0001 [6]). If the request is blockingRequest or the ***Response Type*** parameter is not included, it goes to step Recv-6.0 "Resource handling procedure". If the request is nonBlockingRequestSynch, it goes to step Recv-3.0 "Create <request> resource locally". If the request is nonBlockingRequestAsynch, it goes to step Recv-3.0 "Create <request> resource locally". If the request is flexBlocking, the Receiver CSE shall make the decision to respond using blocking or non-blocking based on its own local context (memory, processing capability, etc.) unless specified further in the resource-specific procedure.

Recv-3.0 "Create <request> resource locally": Refer to clause 7.3.2.2 for details.

Recv-4.0 "Create a successResponse": Refer to clause 7.3.3.12 for details.

Recv-5.0 "Send Response Primitive": Refer to clause 7.3.2.4 for details.

Recv-6.0 "Resource handling procedure": Refer to Figure 7.2.2.2-2 for details.

Recv-7.0 "Update <request> resource": Refer to clause 7.3.2.5 for details. This step is only valid when the request is non-blocking.

Recv-8.0 "Send Notification": Refer to clause 7.5.1.2.5 for details.

Recv-9.0 "Wait for a Response primitive": Refer to clause 7.3.1.3 for details.

Recv-10.0 "Send Response Primitive": Refer to clause 7.3.3.16 for details.

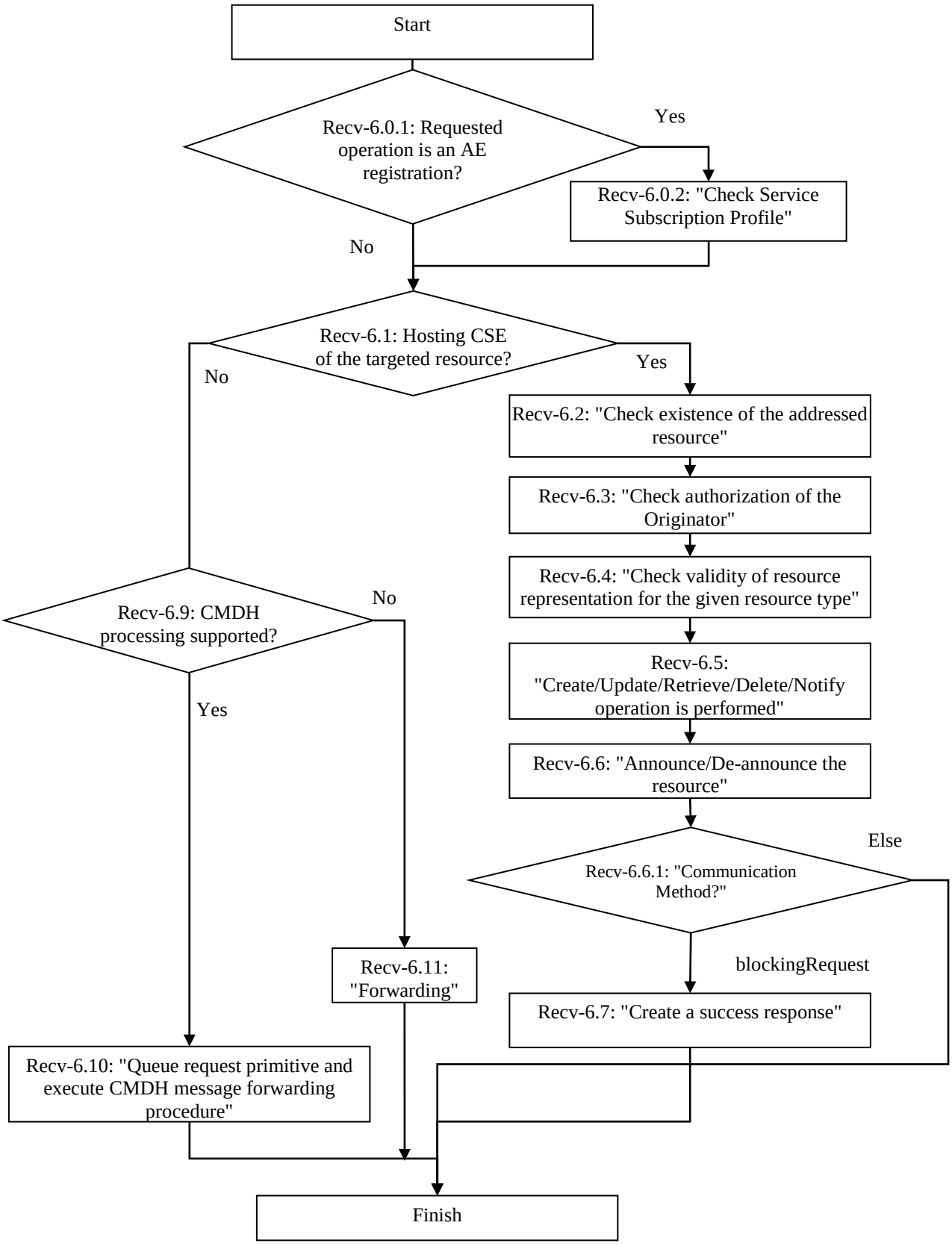

Figure 7.2.2.2-2: Resource handling procedure

Figure 7.2.2.2-2 describes the generic procedure to resource handling procedures.

Recv-6.0.1 "Requested operation is an AE registration?": If the requested operation is an AE registration, then it goes to Recv-6.0.2 "Check Service Subscription Profile". Otherwise, it goes to Recv-6.1.

Recv-6.0.2 "Check Service Subscription Profile": Refer to clause 7.3.2.7 for details.

Recv-6.1 "Hosting CSE of the targeted resource?": The step checks if the receiver is a transit CSE or the Hosting CSE of the received Request by examining the ***To*** parameter of the Request primitive. If the receiver hosts the resource that the address in the ***To*** parameter represents, the receiver is the Hosting CSE (goes to Recv-6.2"Check existence of the addressed resource", Yes branch). Otherwise, the receiver is the Transit CSE (goes to Recv-6.9 "CMDH processing supported?", No branch). Refer to clause 7.3.2.8 for details.

Recv-6.2 "Check existence of the addressed resource": Refer to clause 7.3.3.1 for details.

Recv-6.3 "Check authorization of the Originator": Refer to clause 7.3.3.15 for details.

Recv-6.4 "Check validity of resource representation": Refer to clause 7.3.3.3 and clause 7.3.3.4 for details. Notify is not applicable for this step.

Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed": The step represents five common operations which are "Create the resource (clause 7.3.3.5)", "Retrieve the resource (clause 7.3.3.6)", "Update the resource (clause 7.3.3.7)", "Delete the resource (clause 7.3.3.8)" and "Notify processing (clause 7.3.3.9)".

Recv-6.6 "Announce/De-announce the resource": The step represents two common operations which are "Announce the resource" and "De-announce the resource". Refer to clause 7.3.3.10 and clause 7.3.3.11 for details. Notify is not applicable for this step.

Recv-6.6.1 "Communication method?": The Receiver CSE checks whether a received request is blockingRequest or not by using ***Response Type*** parameter (see detail in clause 8.1.2 in oneM2M TS-0001 [6]). If the request was blockingRequest or ***Response Type*** parameter was not included, it goes to step Recv-6.7 "Create a success response". Otherwise, it goes back to the generic procedure of the receiver (Figure 7.2.2.2-1).

Recv-6.7 "Create a success response": Refer to clause 7.3.3.12 for details.

Recv-6.9 "CMDH processing supported?": This step checks whether the Receiver supports the CMDH processing. If the receiver supports CMDH processing, it goes to Recv-6.10 "Queue request primitive and execute CMDH message forwarding procedure" otherwise, it goes to Recv-6.11 "Forwarding".

Recv-6.10 "Queue request primitive and execute CMDH message forwarding procedure": the Receiver CSE shall queue the received request primitive and execute the "CMDH message forwarding procedure". Refer to clause H.2.4 for details.

Recv-6.11 "Forwarding": carry out message forwarding as defined in clause 7.3.2.6.

7.3 Common operations

7.3.1 Originator actions

7.3.1.1 Compose request primitive

The originator shall compose a Request message that shall be mapped to a specific protocol.

The Request shall include mandatory parameters ***Operation***, ***To***, ***From*** and ***Request Identifier***.

The ***Release Version Indicator*** parameter shall be included unless the primitive is being sent to a Release 1 entity. The Request may include the time related parameters ***Originating Timestamp***, ***Request Expiration Timestamp***, ***Result Expiration Timestamp*** and ***Operation Execution Time***.

The Request may include the other parameters as specified in Table 7.2.1.1-1: Request Primitive Parameters.

When including a resource representation in the request primitive for create and update, the originator 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.

EXAMPLE: Any attributes marked with NP shall not be present in the resource representation for the corresponding request primitive.

7.3.1.2 Send a request to the receiver CSE

The originator shall determine the receiver CSE.

The receiver of the Request shall be the registrar CSE of the originator in case the originator is not IN-CSE.

If the originator is the IN-CSE, the receiver of the Request shall be the CSE whose identifier is the prefix of the ***To*** parameter of the Request.

If this results in no matching CSE, then the request is rejected with a ***Response Status Code*** indicating "NOT_FOUND" error.

If this results in multiple CSEs, the request is rejected with a ***Response Status Code*** indicating "INTERNAL_SERVER_ERROR" error.

7.3.1.3 Wait for response primitive

The originator shall wait for the Response primitive from the receiver that corresponds to the Request primitive that was sent by the originator. Correlation between the Request and the corresponding Response is handled by the transport layer or by ***Request Identifier*** parameter of the primitive.

If no Response 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 Response primitive with a ***Response Status Code*** indicating "REQUEST_TIMEOUT" error was received.

7.3.1.4 Retrieve the <request> resource

When the Originator needs to retrieve information about an associated previously issued non-blocking request, the Originator shall request to Retrieve the attributes of the <request> resource. The Originator shall compose the Request primitive with the following parameters and send the Request to the Receiver CSE. See clauses 7.3.1.1 and 7.3.1.2.

NOTE: The Originator may include optional parameters described in clause 8.1.2 of oneM2M TS-0001 [6].

Table 7.3.1.4-1: Request primitive parameter settings

Parameter Name	Value
Operation	Retrieve.
To	This shall be set to the URI of the <request> resource received in the response (acknowledgement) to the previously issued non-blocking request.
From	Id of the Originator.
Request Identifier	The identifier of this request message.
Content	Optionally includes the name of attributes that needs to be retrieved.
Response Type	blockingRequest.

7.3.2 Receiver CSE actions

7.3.2.1 Check the validity of received request primitive

The validity checking of the message carrying the received request primitive is specified by the protocol mapping Technical Specifications (CoAP binding [22], HTTP binding [23], MQTT binding [24], and Websocket binding [42]).

If the ***Request Expiration Timestamp*** is given in the request and has expired, the Receiver CSE shall reject the request with a "REQUEST_TIMEOUT" ***Response Status Code*** parameter value.

If the ***From*** parameter is not present in the request except AE Create request, the Receiver CSE shall reject the request with a "BAD_REQUEST" ***Response Status Code*** parameter value.

If the received request is communicated within an established Security Association (oneM2M TS-0003 [7]), and

- the Receiver knows that the Registree using the established Security Association is an AE; and
- the Receiver knows the AE-ID(s) of the Registree using the established Security Association; and
- the ***From*** parameter does not match the allowed AE-ID(s) of the Registree using the established Security Association;

then the request shall be rejected with an "ORIGINATOR_HAS_NOT_REGISTERED" ***Response Status Code*** parameter value.

If the received request is communicated within an established Security Association, and:

- the Receiver knows that the Registree using the established Security Association is a CSE; and
- the Receiver knows the CSE -ID of the Registree using the established Security Association; and
- if one of the following applies:
  - the ***From*** parameter is an CSE-ID that matches one of the Receiver's Registree CSE's CSE-ID other than the CSE-ID of the Registree using the established Security Association; or
  - the ***From*** parameter is an CSE-Relative C-Type AE-ID-Stem; or
  - the ***From*** parameter is an SP-Relative AE-ID or Absolute AE-ID with a C-Type AE-ID-Stem, and the CSE-ID portion of the ***From*** parameter matches one of the Receiver's Registree CSE's CSE-ID other than the CSE-ID of the Registree for the established Security Association;

then the request shall be rejected with an "ORIGINATOR_HAS_NOT_REGISTERED" ***Response Status Code*** parameter value.

NOTE: An SP-Relative-AE-ID or Absolute AE-ID with a C-Type AE-ID-Stem always includes a CSE-ID portion (see oneM2M TS-0001 [6]).

If the received request is communicated outside of an established Security Association; and:

- if the ***From*** parameter includes an AE-ID; and
- the request is not a CREATE <AE> Request; and
- the ***From*** parameter does not match the AE-ID of an AE currently registered to the Receiver;

then the request shall be rejected with an "ORIGINATOR_HAS_NOT_REGISTERED" ***Response Status Code*** parameter value.

If the received request is communicated outside of an established Security Association, and the ***From*** parameter includes a CSE-ID, then the request shall be rejected with a "SECURITY_ASSOCIATION_REQUIRED" ***Response Status Code*** parameter value.

If a received request needs to be forwarded to another CSE and if CMDH processing is supported, then in addition, the "CMDH message validation procedure" defined in clause H.2.3 shall be carried out.

If the message is not valid, the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

If ***Resource Type*** is not present or is invalid in a CREATE request, the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

If the ***Filter Criteria*** parameter is included in a CREATE request, the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

If the ***Result Content*** is invalid for a given operation (Refer TS-0001 Table 8.1.2-1: Summary of Result Content Values) then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

If the receiver does not support the content format (i.e. type of serialization) requested by the originator, the request may be rejected with a ***Response Status Code*** indicating "NOT_ACCEPTABLE" error.

If the receiver does not support the content format sent by the originator, the request may be rejected with a ***Response Status Code*** indicating "UNSUPPORTED_MEDIA_TYPE" error.

If the ***Release Version Indicator*** is not present in the request the Receiver CSE shall add the ***Release Version Indicator*** with value set to 1.

If the Receiver CSE is the Hosting CSE and it does not support the release value present in the ***Release Version Indicator*** then the request shall be rejected with a ***Response Status Code*** indicating "RELEASE_VERSION_NOT_SUPPORTED" error.

7.3.2.2 Create <request> resource locally

Creation of a <request> resource can only be done implicitly by a Receiver CSE. When the Receiver CSE receives a request in non-blocking mode (i.e. the ***Response Type*** parameter of the request is set to either "nonBlockingRequestSynch" or "nonBlockingRequestAsynch") targeting any other resource type or requesting a notification, and if the Receiver CSE supports the <request> resource type as indicated by the *supportedResourceType* attribute of the <CSEBase> resource, the Receiver CSE shall create an instance of <request> resource based on the following steps. If the Receiver CSE does not support the <request> resource type, the "nonBlockingRequestSynch" request shall be rejected with a ***Response Status Code*** indicating "NON_BLOCKING_SYNCH_REQUEST_NOT_SUPPORTED" error. In the case of a "nonBlockingRequestAsynch" request, a Receiver CSE that does not support the <request> resource type shall respond to an acceptable request with a response containing an Acknowledgement without a reference to a resource containing the context of the request.

The Receiver CSE of a non-blocking request is the Hosting CSE for the <request> resource that shall be associated with the non-blocking request.

1) Assign values to the common attributes of the <request> resource according to Table 7.3.2.2-1.

Table 7.3.2.2-1: Common attributes settings for <request> resource

Attribute Name	Value
resourceType	Request.
resourceID	Hosting CSE shall assign a value to this attribute.
expirationTime	The value of the *expirationTime* shall be chosen dependent on the ***Request Expiration Timestamp***, ***Result Expiration Timestamp***, ***Operation Execution Time*** and ***Result Persistence*** parameters associated with the original request. If the value consistent with the ***Request Expiration Timestamp***, ***Result Expiration Timestamp***, ***Operation Execution Time*** and ***Result Persistence*** parameters is too long, the Hosting CSE shall reject the request. The *expirationTime* of the <request> resource should last longer than the specified ***Result Persistence*** if provided in the request.
parentID	The parent resource of a <request> resource shall be the <CSEBase> resource of the Hosting CSE.
creationTime	Date/time of creation of this resource.
lastModifiedTime	Date/time which is equal to the *creationTime*.
accessControlPolicyIDs	Populate with the resource identifier of an <accessControlPolicy> that contains the following: In the *privileges* attribute: • Allow RETRIEVE, UPDATE and DELETE operations for the Hosting CSE. • Allow RETRIEVE and DELETE operations for the Originator, i.e. the value of the parameter ***From*** in the associated non-blocking request. In the *selfPrivileges* attribute: 1. Allow UPDATE operations for the Originator, i.e. the value of the parameter ***From*** in the associated non-blocking request.
labels	Originator ID.
resourceName	Hosting CSE shall assign a value to this attribute.

2) Assign values to the resource-specific attributes of the <request> resource according to Table 7.3.2.2-2.

Table 7.3.2.2-2: Resource-specific attributes settings for <request> resource

Attribute Name	Value
operation	The value of the parameter ***Operation*** in the associated non-blocking request.
target	The value of the parameter ***To*** in the associated non-blocking request.
originator	The value of the parameter ***From*** in the associated non-blocking request.
requestID	The value of the parameter ***Request Identifier*** in the associated non-blocking request.
metaInformation	The content of this attribute is set to information in optional parameters described in clause 8.1.2 of oneM2M TS-0001 [6] given in the associated non-blocking request.
content	The value of the parameter ***Content***, if any, in the associated non-blocking request.
requestStatus	The Receiver CSE shall set this to "PENDING".
operationResult	Empty.

7.3.2.3 Create a success response (acknowledgement)

The Receiver CSE shall create a Response primitive. The Receiver CSE shall include the following parameters in the Response primitive.

Table 7.3.2.3-1: Response primitive parameter settings

Parameter Name	Value
Response Status Code	"ACCEPTED"
Request Identifier	The value of the parameter ***Request Identifier*** in the associated non-blocking request.
Originating Timestamp	Timestamp when this message was built.
Content	Reference to the <request> of the associated non-blocking request only if <request> resource is supported.

7.3.2.4 Send response primitive (acknowledgement)

A Response primitive shall be sent back to the originator.

7.3.2.5 Update <request> resource

Changes in the attributes of a <request> resource shall be done by the Hosting CSE implicitly due to changes of the status (*requestStatus*) of the associated non-blocking request or due to the reception of an operation result (*operationResult*) in response to the associated non-blocking request. The Receiver CSE shall update attributes of an instance of the <request> resource based on the following steps.

1) Update values of the common attributes of the <request> resource according to Table 7.3.2.5-1.

Table 7.3.2.5-1: Common attributes settings for <request> resource

Attribute Name	Value
lastModifiedTime	Date/time of the last modification.

2) Update values of the resource-specific attributes of <request> resource according to Table 7.3.2.5-2.

Table 7.3.2.5-2: Resource-specific attributes settings for <request> resource

Attribute Name	Value
requestStatus	If the Receiver CSE is a Transit CSE and the previously received request has been successfully forwarded to the next hop, this shall be set to "FORWARDED". If the Receiver CSE is a Transit CSE and the previously received request has been rejected by the next hop, this shall be set to "FAILED". If the Receiver CSE is the Target CSE (i.e. ***To*** parameter in the request message starts with the CSEBase URI of the Receiver CSE) and the originally requested operation has been completed, this shall be set to "COMPLETED".
operationResult	This shall contain the response message of the originally requested operation – if any – in line with the ***rc*** parameter in the associated non-blocking request.

7.3.2.6 Forwarding

When a receiver CSE is not the Hosting CSE, i.e. the CSE-ID of the receiver CSE is different from the CSE-ID in the ***To*** parameter, the receiver CSE shall attempt to forward the primitive. The Receiver CSE checks each of its <remoteCSE> resources to find whether the CSE-ID in the ***To*** parameter of the primitive matches either the CSE-ID attribute or one of the CSE-IDs in the *descendantCSEs* attribute of the <remoteCSE>.

- If a match is found, the CSE shall retarget the request to the *pointOfAccess* of the matching <remoteCSE> resource.
- If a match is not found, and the CSE received the request from an AE or a descendant CSE, and the CSE is not the IN-CSE, then it shall retarget the request to its Registrar CSE.
- If a match is not found and the CSE is the IN-CSE, then the CSE shall not forward the request and it shall respond with an error using ***Response Status Code*** "NOT_FOUND".
- If a match is not found and the CSE is not the IN-CSE and the CSE receives the request from its registrar CSE, then the CSE shall not forward the request and it shall respond with an error using ***Response Status Code*** "NOT_FOUND".

When the receiver CSE forwards the primitive and the ***From*** parameter value has CSE-relative-ID format, the receiver shall convert the ID format of the ***From*** parameter into SP-relative by pre-pending its own CSE-ID. If the receiver CSE is the IN-CSE and the ***To*** parameter targets another SP domain, the IN-CSE shall convert the ID format of the ***From*** parameter into Absolute format by pre-pending another SP-ID.

If any of the ***Request Expiration Timestamp, Result Expiration Timestamp*** or ***Operation Execution Time*** parameters are set in the request, the receiver CSE should forward the request before the earliest of these times. If the any of the timestamps are in the past, it shall reject the request with a "REQUEST_TIMEOUT" ***Response Status Code*** parameter value and not forward the request.

A receiver CSE shall remove the ***Release Version Indicator*** and ***Vendor Information*** from the request or response before retargeting a primitive to a Release 1 entity.

A receiver CSE shall indicate the content serialization to be used in a response in the retargeted primitive.

Acting as an originator the CSE shall perform the following procedures:

1) "Send a Request to the receiver CSE". Refer to clause 7.3.1.2 for details.
2) "Wait for Response primitive". Refer to clause 7.3.1.3 for details.

When the Response is received the receiver CSE shall:

1) Primitive specific procedure: Forward the Response to the original CSE.

7.3.2.7 Check Service Subscription Profile

In order to validate whether or not the AE registration request complies with the subscriber's service subscription profile, the Receiver shall check if a <serviceSubscribedNode> child resource of the subscriber's <m2mService SubscriptionProfile> resource exists, with a CSE-ID attribute that matches the Receiver owned CSE-ID. If this condition is met, the Receiver shall further check whether the Registree AE is included in the linked (i.e. *ruleLinks* attribute) <serviceSubscribedAppRules> resource(s).

7.3.2.8 Check Hosting CSE of the targeted resource

The Receiver shall check the ***To*** parameter of the request, depending upon the addressing modes in the request. The following handling shall be done:

***To* parameter with Absolute Resource ID representation:**

If the ***To*** parameter in the request starts with M2M-SP-ID (i.e. Absolute Resource ID representation) but M2M-SP-ID in the ***To*** parameter is different from M2M-SP-ID of the receiver, then receiver is a transit CSE.

If the ***To*** parameter in the request starts with M2M-SP-ID (i.e. Absolute Resource ID representation) of the receiver, but the CSE-ID in the ***To*** parameter is different from the CSE-ID of the receiver, then the receiver is a transit CSE.

If the ***To*** parameter in the request starts with M2M-SP-ID (i.e. Absolute Resource ID representation) of the receiver, and the CSE-ID in the ***To*** parameter is same as CSE-ID of the receiver, then the receiver is the Hosting CSE.

***To* parameter with SP-Relative Resource ID representation:**

If the ***To*** parameter in the request starts with CSE-ID (i.e. SP-Relative Resource ID representation) of the receiver, and the CSE-ID in the ***To*** parameter is different from CSE-ID of the receiver, then the receiver is a transit CSE.

If the ***To*** parameter in the request starts with CSE-ID (i.e. SP-Relative Resource ID representation) of the receiver, and the CSE-ID in the ***To*** parameter is same as CSE-ID of the receiver, then the receiver is the Hosting CSE.

***To* parameter with CSE-Relative Resource ID representation:**

If the ***To*** parameter in the request does not start with CSE-ID (i.e. CSE-Relative Resource ID representation), then the receiver is the Hosting CSE.

When the receiver is a transit CSE:

- If the request is an AE/CSE registration request, then the request is rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.
- If the transit CSE is not able to receive asynchronous messages from the next-hop CSE, it shall set the ***Response Type*** in the forwarded request to blockingRequest, nonBlockingRequestSynch or flexBlocking without notification targets.
- Either CMDH Message Forwarding (Recv-6.9) or Forwarding (Recv-6.10) shall apply as depicted in Figure 7.2.2.2-2 Resource handling procedure except for AE/CSE registration request.

7.3.3 Hosting CSE actions

7.3.3.1 Check supported resource types

If the request is a valid CREATE request, but the Hosting CSE does not implement the requested resource type, then the Hosting CSE shall reject the request and return an error response with a ***Response Status Code*** indicating "NOT IMPLEMENTED".

7.3.3.2 Check existence of the addressed resource

If the ***Request Expiration Timestamp*** is given in the request and has expired, the Hosting CSE shall reject the request with a "REQUEST_TIMEOUT" ***Response Status Code*** parameter value. Otherwise, the Hosting CSE should handle the request before the time specified in ***Request Expiration Timestamp.***

The Hosting CSE shall check if the resource addressed by the ***To*** parameter exists in the repository. If the resource does not exist, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

The Hosting CSE shall also check the existence of target resources based on conditions specified in the ***Filter Criteria*** parameter in the Retrieve/Update/Delete operation. If there are no matching target resources, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

If the Hosting CSE does not support the content format (i.e. type of serialization) requested by the originator, the request shall be rejected with a ***Response Status Code*** indicating "NOT_ACCEPTABLE" error.

If the Hosting CSE does not support the content format sent by the originator, the request shall be rejected with a ***Response Status Code*** indicating "UNSUPPORTED_MEDIA_TYPE" error.

7.3.3.3 Check validity of resource representation for CREATE

The handling below shall apply 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 the CREATE request has a Resource Type that is not listed in the child resource tables, defined in clause 7.4 corresponding to the addressed resource, then the request shall be rejected with a ***Response Status Code*** indicating "INVALID_CHILD_RESOURCE_TYPE" error.

If no resource representation is present in the CREATE request, then the request is rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

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 ***Response Status Code*** indicating "BAD_REQUEST" error.

There are three cases where the Hosting CSE shall configure or override an *expirationTime* value that differs from the value specified in the resource representation (if present):

1) The Originator does not specify an *expirationTime.*

2) The Originator requests an expiration time that is later than the *expirationTime* of the parent.

3) The Hosting CSE determines that the expiration time requested by the Originator does not meet its requirements (e.g. based on a local policy).

In each of these cases, the Hosting CSE shall configure an *expirationTime* into the resource that is less than or equal to the *expirationTime* of the parent resource. In addition, the Hosting CSE shall communicate the modified value back to the originator in the response if the ***Result Content*** parameter permits this.

If the *creator* attribute is present in the resource representation and supported by the type of resource to be created its value shall be NULL. If the originator provides a value for the *creator* attribute but resource type does not support the *creator* attribute, then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error. If the originator provides a value for the *creator* attribute within the request, the Hosting CSE shall reject the request with a Response Status Code indicating "BAD_REQUEST" error.

The resource descriptions in clause 7.4 include tables that specify the attributes of each resource and the optionality of the attribute in a CREATE or UPDATE request, see clause 7.4.1.1. A request containing an attribute not listed in the table shall be rejected with a "BAD_REQUEST" error.

M attribute for create request

If the attribute is present in the resource representation in the CREATE request, the Hosting CSE shall check if the value is acceptable according to internal policies.

If the provided value is not accepted, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

If the attribute is not present in the resource representation in the CREATE request the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

O attribute for create request

If the attribute is present in the resource representation in the CREATE request, the Hosting CSE shall check if the value is acceptable according to internal policies.

If the provided value is not accepted then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

NP attribute for create request

If the attribute is present in the resource representation in the CREATE request, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

7.3.3.4 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 ***Response Status Code*** indicating "BAD_REQUEST" error.

If the *expirationTime* attribute is present in the UPDATE request, and its value is earlier than the value of the *expirationTime* attribute that it is updating, then the Hosting CSE shall check if the targeted resource has any child resource whose *expirationTime* attribute value is later than the *expirationTime* value in the UPDATE request. If yes, the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

The resource descriptions in clause 7.4 include tables that specify the attributes of each resource and the optionality of the attribute in a CREATE or UPDATE request, see clause 7.4.1.1. A request containing an attribute not listed in the table shall be rejected with a "BAD_REQUEST" error.

O attribute for update request

If the attribute is present in the resource representation in the UPDATE request, the Hosting CSE shall check if the value is acceptable according to internal policies.

If the provided value is not accepted, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

NP attribute for update request

If the attribute is present in the resource representation in the UPDATE request, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

7.3.3.5 Create the resource

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at the time and shall not perform the procedures before the time.

A new resource shall be created with the ***lastModifiedTime*** attribute of the resource set to the same value as the ***creationTime*** attribute of the resource.

The following rules shall be applied:

- The structured resource identifier of the created resource shall be the identifier of its parent resource with the *resourceName* appended. (e.g. myCSE/myContainer, for an application resource with *resourceName* "myContainer" created in the parent resource myCSE).

- When configuring the *resourceName* attribute of the new resource, the Hosting CSE shall use the name provided in the *resourceName* attribute within the content of the request. The Hosting CSE shall first check for the presence of any resources having a *resourceName* attribute that matches the one specified in the request and that have the same parent as the new resource being created. If such a resource exists, then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "CONFLICT" error. If the *resourceName* is not provided in the request, the Hosting CSE shall generate and assign a name to the *resourceName* attribute of the new resource.

- If the *expirationTime* attribute is present in the resource representation of the resource that is to be created and the *expirationTime* is set to a non-negative time, then an expiration timer shall be started by the Hosting CSE. At timer expiration the related resource is deleted as specified in "Delete the addressed resource".

- Attributes of the parent resource shall be updated, if applicable. For example, the *currentByteSize* attribute of a <container> resource will be updated upon child <contentInstance> resource creation. An attribute update of a parent resource is resource type specific, as specified in clause 7.4.

- If the *creator* attribute is present in the resource representation, and is supported by the type of resource to be created, and is NULL, then the Hosting CSE shall include the *creator* attribute in the resource to be created. The Hosting CSE shall assign the *creator* attribute with a value equal to the value carried in the ***From*** request parameter. If the *creator* attribute is not present in the resource representation of the request, the Hosting CSE shall not include the *creator* attribute in the resource to be created.

- The Hosting CSE shall check if the created resource references an Application Entity Resource ID. If so the Hosting CSE shall send a Notify request primitive to the IN-CSE, requesting to add the entry to the <AEContactList> resource.

For setting the attributes in the resource representation the following rules shall apply in CREATE request primitives:

M attribute for create request

If the provided value is acceptable, the Hosting CSE shall use the provided value in the resource representation of the created resource.

O attribute for create request

If a value is provided and accepted, then the Hosting CSE shall use the provided value in the resource representation of the created resource.

If the attribute is not provided or accepted, but the multiplicity of the attribute is "1" in the resource, the Hosting CSE shall assign default value or assign value based on local policy, or the value of specified in clause 7.4.

If the attribute is not present in the resource representation in the CREATE request and the multiplicity of the attribute is "0..1" in the resource, the Hosting CSE shall create the resource without the attribute unless otherwise specified in resource type-specific procedures defined in clause 7.4.

NP attribute for create request

If the attribute is not present in the resource representation in the CREATE request, and the multiplicity of the attribute is "1" in the resource, then the Hosting CSE shall create the resource with the default value.

7.3.3.6 Retrieve the resource

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at the time and shall not perform the procedures before the time.

When the resource is read to provide a response to Retrieve request primitives:

Full retrieve request: the request target is a resource given in the ***To*** parameter

The content of the Response to the Retrieve Request shall comply to the ***Result Content*** parameter in the Request. If a ***Result Content*** is not provided in the Request, the representation of the resource which includes all the attributes shall be returned.

Partial retrieve request: there are two cases:

Case 1) the request target is a resource given in the ***To*** parameter and specific attribute names are provided in the ***Content*** parameter:

The values of the resource attribute(s) provided in the ***Content*** parameter shall be retrieved.

Case 2) the request target is a resource given in the ***To*** parameter, the resource attribute is provided in the To parameter as a fragment identifier component of URI following '#' character [2]. The resource attribute shall be represented as a short name and shall belong to short name list in Table 8.2.3-1 to Table 8.2.3-5.

The resource attribute provided in the ***To*** parameter shall be retrieved.

7.3.3.7 Update the resource

If the ***Operation Execution Time*** parameter is given in the request, the Hosting CSE should perform the following procedures at that time and shall not perform the procedures before that time.

The Hosting CSE shall check to see if the target resource has a child <subscription> resource with *notificationEventType* attribute set to "Blocking_Update" according to the procedure specified in clause 7.5.1.2.19.

Attributes that are not included in the ***Content*** parameter of the addressed resource shall not be changed by the Hosting CSE. For attributes provided in the ***Content*** parameter, their content shall be updated while the following rules apply:

If the *announceTo* attribute or *announcedAttribute* attribute of the resource is requested to be updated, the Hosting CSE shall update the attribute as described in the "announce the resource or attribute" and "de-announce the resource or attribute" procedures as specified in the clause 7.3.3.10 and clause 7.3.3.11, respectively.

The Hosting CSE shall check if the update causes a change to a reference to an Application Entity Resource ID. If so the Hosting CSE shall send a Notify request primitive to the IN-CSE, requesting to update the entry to the <AEContactList> resource.

O attribute for update request

If an attribute value is provided and the value is accepted, the Hosting CSE shall use the provided value in the resource representation of the updated resource.

If the attribute is not provided, but the attribute exists in the target resource, the Hosting CSE shall leave the value of that attribute unchanged.

If this attribute is provided in the ***Content*** parameter and does not exist in the target resource, the Hosting CSE shall create such attribute with the provided value.

If this attribute is set to NULL in the ***Content*** parameter and exists in the target resource the following shall apply:

- If the multiplicity of the attribute is shown as "1" or "1(L)" in oneM2M TS-0001 [6], the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "BAD_REQUEST".
- If the multiplicity of the attribute is shown as other than "1" or "1(L)" in oneM2M TS-0001 [6], the Hosting CSE shall delete the attribute.

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 CSE. At timer expiration the related resource is deleted as specified in "Delete the addressed resource".

NP attribute for update request

If the update is successful, the Hosting CSE shall set the *lastModifiedTime* to the current time and the Hosting CSE shall increment the *stateTag* if present.

7.3.3.8 Delete the resource

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at that time and shall not perform the procedures before that time.

The addressed resource with all its attributes shall be deleted. Any expiration timer shall be stopped. This same procedure shall be invoked (recursively) for each child resource of the deleted resource in case the child resource is only linked to the deleted resource.

The Hosting CSE shall check if the deleted child resource leads to changes in its parent resource's attribute(s) (e.g. *currentNrOfInstances*, *currentByteSize* etc.*)*, then these attribute(s) shall be updated.

If the resource is announced, the CSE shall try to de-announce the resource correspondingly.

If the deleted resource had a reference to an Application Entity Resource ID, the Hosting CSE shall send a Notify request primitive to the IN-CSE, requesting to delete the entry from the <AEContactList> resource.

7.3.3.9 Notify processing

7.3.3.9.1 Notify processing when ***To*** parameter is an <AE> resource

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at the time and shall not perform the procedures before the time.

When the Hosting CSE receives a Notify request primitive targeting (i.e. ***To*** parameter) its <AE> resource, the Hosting CSE re-targets the primitive to the AE if the <AE> resource does not have any <pollingChannel> resource as a child.

1) Get the *pointOfAccess* attribute value of the corresponding <AE> resource. If there is no available *pointOfAccess* address then the Hosting CSE shall send the Notify response primitive with a ***Response Status Code*** indicating "TARGET_NOT_REACHABLE" error.

2) Forward the Notify request primitive to the first address retrieved from the *pointOfAccess* value.

3) If the forwarding fails due to "Target not reachable", iterate 2) with the next address.

4) If the Hosting CSE cannot forward it in the end, then it sends a Notify response primitive with a ***Response Status Code*** indicating "TARGET_NOT_REACHABLE" error.

7.3.3.9.2 Notify processing when ***To*** parameter is the <CSEBase> resource

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at the time and shall not perform the procedures before the time.

When the Hosting CSE receives a Notify request primitive targeting (i.e. ***To*** parameter) its <CSEBase> resource, the Hosting CSE shall process the ***Content*** of the Notify request primitive. See clause 7.5.1.

7.3.3.10 Announce the resource or attribute

If a CREATE Request that contains an *announceTo* attribute is received:

1) Compose the CREATE Request primitive as follows:

   - The *link* attribute is set to the URI of the original resource.
   - If the *accessControlPolicyIDs* attribute of the original resource is not present, the *accessControlPolicyIDs* attribute is set to the same value as the parent resource or is set using the local policy of the original resource.
   - Attributes marked with MA in oneM2M TS-0001 [6]. Such attributes shall be included if present in the original resource and set to same value as in the original resource.
   - Attributes marked with OA that are included in the *announcedAttribute* attribute. Such attributes shall be included if present in the original resource and set to same value as in the original resource.
   - The *resourceType* attribute is set to the announced variant of the original resource (see Table 6.3.4.2.1-1).

2) Perform the following steps for each item (announcement target) in the *announceTo* attribute list:

a. If the announcement target is a CSE-ID, check if the CSE hosting the original resource has registered to the corresponding announcement target CSE, or is the registrar CSE for the announcement target and a <remoteCSE> resource for the Host CSE exists on the announcement target CSE.

- If yes, announce the original resource by sending a CREATE Request to the CSE specified in the *announceTo*, addressed to the <remoteCSE> of the CSE hosting the original resource.
- If no, then check if the CSE hosting the original resource has been announced to the announcement target CSE and created a <remoteCSEAnnc> resource as a child of the <CSEBase>
  - If yes, announce the original resource by sending a CREATE Request to the CSE specified in the *announceTo*, addressed to the <remoteCSEAnnc> resource.
  - If no, then CSE hosting the original resource shall perform the following steps:
    - Announce itself to the CSE specified in the *announceTo* attribute such that its <remoteCSEAnnc> resource is present at the announcement target CSE.
    - Send a CREATE Request to the CSE specified in the *announceTo* of the request, addressed to the <remoteCSEAnnc> resource.

b. If the announcement target is not a CSE-ID send a CREATE Request to the CSE represented by the exact URI in the *announceTo* of the request.

c. Wait for the Response to the CREATE that was sent in step a or b.

d. Add the URI of the successfully-announced resource to the *announceTo* attribute of the resource.

3) Include the updated *announceTo* attribute in the ***Content*** parameter of the Response to the received CREATE Request.

If an UPDATE Request that adds a URI or CSE-ID into the *announceTo* attribute is received:

1) Compose the CREATE Request primitive as follows:
   - The *link* attribute is set to the URI of the original resource.
   - If the *accessControlPolicyIDs* of the original resource is not present, the *accessControlPolicyIDs* is set to the same value as the parent resource or is set using the local policy of the original resource.
   - Attributes marked with MA and attributes marked with OA that are included in the *announcedAttribute* attribute. Such attributes shall be present in the original resource and set to same value as the original resource.
2) Perform the following steps for each new item (announcement target) that has been added to the *announceTo* attribute list:
   a. If the announcement target is a CSE-ID, check if the parent resource of the original resource has been announced to the corresponding announcement target CSE.
      - If yes, announce the original resource by sending a CREATE Request to the CSE specified in the *announceTo, addressed to* the announced parent resource.
      - If no, check if the CSE hosting the original resource has registered and created a <remoteCSE> resource to the announcement target CSE:
        - If yes, announce the original resource by sending a CREATE Request to the CSE specified in the *announceTo*, addressed to the <remoteCSE> of the CSE hosting the original resource.
        - If no, then check if the CSE hosting the original resource has been announced to the announcement target CSE and created a <remoteCSEAnnc> resource as a child of the <CSEBase>.

- If yes, announce the original resource by sending a CREATE Request to the CSE specified in the *announceTo*, addressed to the <remoteCSEAnnc> resource.
- If no, then CSE hosting the original resource shall perform the following steps:
    - Announce itself to the CSE specified in the *announceTo* attribute such that its <remoteCSEAnnc> resource is present at the announcement target CSE.
    - Send a CREATE Request to the CSE specified in the *announceTo* of the request, addressed to the <remoteCSEAnnc> resource.

b. If the announcement target is not a CSE-ID, send a CREATE Request to the CSE represented by the exact URI in the *announceTo* of the request.

c. Wait for the Response to the CREATE that was sent in step a or b.

d. Add the URI of the successfully-announced resource to the *announceTo* attribute of the resource.

3) Include the updated *announceTo* attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

If an UPDATE Request that adds the attribute name into the *announcedAttribute* attribute is received:

1) Compose the UPDATE Request primitive. The UPDATE Request shall provide the attribute name for the attribute to be announced, and the initial value for the attribute in the ***Content*** parameter. The initial value shall be the same with the value from the original resource. The attribute that will be announced shall be marked as OA.

2) Send UPDATE Requests to all announced resources listed in the *announceTo* attribute.

3) Wait for Response primitive.

4) Add the attribute name of the successfully announced attribute to the *announcedAttribute* attribute.

5) Include updated *announcedAttribute* attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

If an attribute(s) specified as MA (see oneM2M TS-0001 [6]) or an attribute(s) included in the *announcedAttribute* attribute is updated:

1) Compose an UPDATE Request primitive by including the updated attribute(s) with its associated updated value.

2) Send the UPDATE Request to all CSE(s) represented by the URI(s) in the *announceTo* attribute of the original resource.

3) Wait for the Response primitive(s).

7.3.3.11 De-announce the resource or attribute

If an UPDATE Request that deletes the URI from the *announceTo* attribute is received:

1) Compose the DELETE Request primitive.

2) Send a DELETE Request to the CSE(s) represented by URI(s) in the *announceTo* attribute of the resource, which is not included in the *announceTo* of the request. The ***To*** parameter in the DELETE Request shall be set to the URI for the announced resource that will be deleted.

3) Wait for Response primitive.

4) Remove the URI of successfully de-announced resource from the *announceTo* attribute of the resource.

5) Include updated *announceTo* attribute in the ***Content*** parameter in the Response to the UPDATE Request of the original resource.

If a DELETE Request is received:

1) Compose the DELETE Request primitive.

2) Send DELETE Requests to all of the CSE(s) represented by the URI(s) in the *announceTo* attribute of the resource.

3) Wait for Response primitive.

If an UPDATE Request that deletes the attribute name from the *announcedAttribute* attribute is received:

1) Compose the UPDATE Request primitive. The ***To*** parameter in the UPDATE Request shall be set to the URI for the announced resource. The UPDATE Request shall set the attribute that will be de-announced (i.e. to be deleted) in the ***Content*** parameter to NULL. The attribute that will be de-announced shall be marked as OA.

2) Send UPDATE Requests to all announced resources listed in the *announceTo* attribute of the original resource.

3) Wait for Response primitive.

4) Delete the attribute name of the successfully de-announced attribute from the *announcedAttribute* attribute.

5) Include updated *announcedAttribute* attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

If an attribute(s) specified as MA (see oneM2M TS-0001 [6]) or an attribute(s) included in the *announcedAttribute* attribute is deleted:

1) Compose an UPDATE Request primitive by including the deleted attribute(s) with its value set to NULL.

2) Send the UPDATE Request to all CSE(s) represented by the URI(s) in the *announceTo* attribute of the original resource.

3) Wait for Response primitive.

4) Delete the attribute name of the successfully de-announced attribute from the *announcedAttribute* attribute if it was present.

7.3.3.12 Create a success response

The Hosting CSE shall create a success response primitive with a ***Response Status Code*** indicating:

- "CREATED" in case of Create operation;
- "OK" in case of Retrieve operation;
- "UPDATED" in case of Update operation;
- "DELETED" in case of Delete operation; and
- "OK" in case of Notify operation.

The Hosting CSE shall include ***Request Identifier*** parameter in the response primitive. The ***Release Version Indicator*** parameter shall be included unless the primitive is being sent to a Release 1 entity.

The Hosting CSE shall include the ***Content*** parameter in a Retrieve Response. The Hosting CSE shall include the ***Content*** parameter in a Create/Update/Delete Response unless the ***Result Content*** is set to 0 (nothing). If the ***Result Content*** is not given in the Request, the default value for Delete is 0 (nothing), for Create/Retrieve/Update it is 1 (all attributes).

The information of the ***Content*** parameter shall comply to the value of the ***Result Content*** request parameter of the corresponding Request.

More details can be found in clause 7.2.1.2 (Response primitive format).

The response content serialization shall use the type indicated in the received request. If a content serialization cannot be determined from the request then the serialization of the response shall use one of the serialization types specified in the *contentSerialization* attribute of the originator.

The Hosting CSE may include ***To***, ***From***, ***Originating Timestamp***, ***Result Expiration Timestamp***, ***Event Category*** parameters.

7.3.3.13 Create an error response

The receiver shall create an error response primitive with a ***Response Status Code*** indicating the detected error condition.

The response content serialization shall use the type indicated in the received request. If a content serialization cannot be determined from the request then the serialization of the response shall use one of the serialization types specified in the *contentSerialization* attribute of the originator.

NOTE: Possible error codes and error handling is described in resource specific procedures.

7.3.3.14 Resource discovery procedure

If the ***Operation Execution Time*** is given in the request, the Hosting CSE should perform the following procedures at the time and shall not perform the procedures before the time.

Resource discovery is used to discover resources in a CSE. A Resource discovery request is done by sending a Retrieve request with ***filterUsage***, one of the ***Filter Criteria*** parameters, configured as "Discovery" and the request may include other ***Filter Criteria*** parameters as well. A resource discovery request procedure shall be comprised of the following actions.

Originator:

The Originator shall follow the steps from Orig-1.0 to Orig-6.0 specified in clause 7.2.2.1 Generic Resource Request Procedure for Originator.

In addition to Orig-1.0, the following steps shall be performed.

The ***To*** parameter in the Retrieve Request indicates the root of where the discovery begins.

The Retrieve Request shall include a ***Filter Criteria*** request parameter that includes a ***filterUsage*** element configured with either "Discovery Criteria" or "IPE On-demand Discovery".

The Retrieve Request may include other elements of ***Filter Criteria***.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-7.0 specified in clause 7.2.2.2.

The Hosting CSE shall not perform steps from Recv-6.3 to Recv-6.6 but perform the following steps instead.

The Receiver shall find the resources that match all the configured ***Filter Criteria*** and to which the Originator has "Discover" privilege, among all the children/descendent resource of the addressed resource. As part of this search, the Receiver will not consider any child/descendent <AE> resource with *registrationStatus* attribute set to INACTIVE, and any child/descendent resources of this INACTIVE <AE> resource.

If the addressed resource is an <AE> resource representing the IPE by its *labels* attribute, the Hosting CSE shall find resources using the ***Filter Criteria***. When the Hosting CSE finds no match, the Hosting CSE shall check the ***filterUsage*** element. If the ***filterUsage*** element is set to "IPE On-demand Discovery", then the Hosting CSE shall send the Notify request to the IPE to trigger the external discovery procedure (see clause 7.5.1.2.8 for more details). If the Hosting CSE receives a successful Notify response, the Hosting CSE shall find resources among the resources on the Hosting CSE listed in the Notify response using the ***Filter Criteria*** and check the Originator's "Discover" privilege. If the Hosting CSE receives an unsuccessful Notify response from the IPE, then the Hosting CSE shall use the same ***Response Status Code*** in the response to the Originator.

In Recv-6.7, the Receiver shall include addresses for all the found resources in the CSE-relative resource identifier format.

The Receiver shall perform Recv-6.8 and the procedure is terminated.

7.3.3.15 Check authorization of the originator

If the target resource contains the *accessControlPolicyIDs* attribute, the Hosting CSE shall use the linked <accessControlPolicy> resources as in the evaluation procedure below. See clause 9.6.1.3.2 in oneM2M TS-0001 [6] for how to handle the case where the target resource has no accessControlPolicyIDs attribute.

The evaluation procedure shall be performed as following:

1) The Hosting CSE retrieves the access control rules from *privilege* attribute of the <accessControlPolicy> which is linked as the *accessControlPolicyIDs*. If the target is <accessControlPolicy> resource, it retrieves the rules from *selfPrivilege* attribute instead.

2) The Hosting CSE checks the following conditions for the access control rules. If there is any rule satisfying all conditions then the evaluation is successful, otherwise access is denied. For more details, see clause 7.1.5 in oneM2M TS-0003 [7].

   - *accessControlOriginators* of the rule includes the Originator information. The accessControlOriginators parameter comprises a list of domain, CSE-IDs, AE-IDs, the resource-ID of a <group> resource that contains <AE> or <remoteCSE> as member or Role-ID. The accessControlOriginators parameter can be set to reserved keyword "all" to grant access to all originators. It is allowed to include the wildcard character, "*", into the URI string of domain, CSE-ID and AE-ID at any level. See clause 9.6.2.1 in oneM2M TS-0001 [6].

Table 7.3.3.15-1: Types of Parameters in *accessControlOriginators*

Name	Description	Wildcard applicability
domain	A M2M-SP-ID representing domain	Allowed
originatorID	CSE-ID	Allowed
	AE-ID	Allowed
group	The resource-ID of a <group> resource which contains <AE> or <remoteCSE> as member	Not allowed
all	Any Originators are allowed	Not allowed
Role-ID	A Role Identifier as defined in clause 7.1.14 of oneM2M TS-0001 [6]	Not allowed

   - *accessControlContexts* of the rule includes the request context, if the rule includes the *accessControlContexts.*
     - If the accessControlOriginators includes a groupID, the Hosting CSE checks if the Originator is a member of that group resource.
   - *accessControlOperations* of the rule matches the operation type of the request.
   - If the *accessControlAuthenticationFlag* is true, then access control rule applies only if the Originator is considered to be authenticated by the Hosting CSE according to clause 7.1.2 in oneM2M TS-0003 [7].

If the evaluation of these access control rules results in access being permitted, the authorization check process ends. If the result is "DENY", the Hosting CSE proceeds with Dynamic Authorization (see clause 7.3 in oneM2M TS-0003 [7]) if it supports that.

If Dynamic Authorization returns "PERMIT", the authorization check process ends. If the result is "DENY" the Hosting CSE proceeds with Distributed Authorization (see clause 7.5 in oneM2M TS-0003 [7]) if it supports that. If this returns "PERMIT", the authorization check process ends.

If the Hosting CSE reaches the point where it has tried all the authorization processes that it supports and they have all returned "DENY" the Hosting CSE shall reject the request with an "ORIGINATOR_HAS_NO_PRIVILEGE" ***Response Status Code*** parameter value.

7.3.3.16 Send response primitive

A Response primitive shall be sent back to the Originator. If the primitive is successful response and the ***Result Expiration Timestamp*** is given in the request, then the Hosting CSE or Transit CSE should send the primitive before the ***Result Expiration Timestamp***.

7.3.3.17 Using Filter Criteria for identification of target resources

7.3.3.17.0 Introduction

When the ***Filter Criteria*** primitive parameter is present in a request primitive, it shall be applied for identification of the applicable target resources of the respective operation. This may apply to Retrieve, Delete, Discovery and Semantic Resource Discovery operations as specified in clauses 7.3.3.6, 7.3.3.8, 7.3.3.14 and 7.3.3.18 respectively.

The ***Filter Criteria*** primitive parameter defines matching conditions on resource attributes and filter handling conditions. Matching conditions are evaluated against resources and, when true, determine the matched resources. The filter handling conditions provide additional input applied to the matched resource set to determine the filtering result (e.g. maximum number of resources to be included in the filtering result). The filtering result may be composed of one or more resources and shall be used as the target of the operation. Table 7.3.3.17.0-1 summarizes the various filter criteria and conditions. Each row in the table represents a different filter condition type.

If multiple matching conditions of the same type (i.e. same condition tag) are present in the ***Filter Criteria*** parameter, these shall be combined by applying logical OR operation. This applies to the condition tags ***labels***, ***resourceType***, ***contentType*** or ***attribute*** with multiplicity n > 1.

If multiple matching conditions of different type (i.e. different condition tags) are present in the ***Filter Criteria*** parameter, then the combined condition shall be derived by applying the logical operation specified by the *filterOperation* condition. By default logical AND operation shall be used if the *filterOperation* condition is not present.

EXAMPLE:

1. ***labels***=floor1, ***stateTagSmaller***=3 will match if both conditions are true [default AND when ***filterOperation*** is not specified]
2. ***labels***=floor1, ***stateTagSmaller***=3, ***filterOperation***=1 will match if both conditions are true
3. ***labels***=floor1, ***stateTagSmaller***=3, ***filterOperation***=2 will match if either condition is true
4. ***labels***=floor1, ***labels***=floor2, ***filterOperation***=1 will match if either condition is true [***filterOperation*** has no effect when all condition tags are the same]
5. ***labels***=floor1, ***stateTagSmaller***=3, ***labels***=floor2, ***filterOperation=2*** will match if any of these conditions are true resource has [labels with value "floor1" OR "floor2"] OR stateTagSmaller than 3

Table 7.3.3.17.0-1: Summary on Filter conditions

Condition Tag	Multiplicity	Targeted Resource Attribute	Matching Condition
createdBefore	0..1	creationTime	creationTime < createdBefore, see clause 7.3.3.17.1
createdAfter	0..1		createdAfter ≤ creationTime, see clause 7.3.3.17.1
unmodifiedSince	0..1	lastModifiedTime	lastModifiedTime < unmodifiedSince, see clause 7.3.3.17.2
modifiedSince	0..1		unmodifiedSince ≤ lastModifiedTime, see clause 7.3.3.17.2
stateTagSmaller	0..1	stateTag	stateTag < stateTagSmaller, see clause 7.3.3.17.3
stateTagBigger	0..1		stateTagBigger ≤ stateTag, see clause 7.3.3.17.3
expireBefore	0..1	expirationTime	expirationTime < expireBefore, see clause 7.3.3.17.4
expireAfter	0..1		expireAfter ≤ expirationTime, see clause 7.3.3.17.4
labels	0..1	labels	see clause 7.3.3.17.5
childLabels	0..1		see clause 7.3.3.17.5
parentLabels	0..1		see clause 7.3.3.17.5
resourceType	0..1	resourceType	see clause 7.3.3.17.6
childResourceType	0..1		see clause 7.3.3.17.6
parentResourceType	0..1		see clause 7.3.3.17.6

Condition Tag	Multiplicity	Targeted Resource Attribute	Matching Condition
sizeBelow	0..1	contentSize	contentSize < sizeBelow, see clause 7.3.3.17.7
sizeAbove	0..1		sizeAbove ≤ contentSize, see clause 7.3.3.17.7
typeOfContent	0..n	contentInfo	matched with typeOfContent component in contentInfo, see clause 7.3.3.17.8
attribute	0..n	(variable)	name and value of Filter Criteria attribute matches resource attribute, see clause 7.3.3.17.9
childAttribute	0..n	(variable)	name and value of Filter Criteria attribute matches resource attribute, see clause 7.3.3.17.9
parentAttribute	0..n	(variable)	name and value of Filter Criteria attribute matches resource attribute, see clause 7.3.3.17.9
limit	0..1	(not applicable)	Constraint on maximum number of targeted resources, see clause 7.3.3.17.10
filterUsage	0..1	(not applicable)	Indicator specifying the use case of Filter Criteria parameters
semanticsFilter	0..n	(not applicable)	Filtering conditions expressed in SPARQL [i.6]. These are applicable to the descriptor attribute of <semanticDescriptor> children associated with discoverable resources. When multiple semanticsFilter elements are provided, the matching condition is fulfilled if any of the individual conditions is matched
filterOperation	0..1	(not applicable)	Indicates the logical operation (AND/OR) to be used for different conditions. The default value is logical AND
contentFilterSyntax	0..1	(not applicable)	Indicates the Identifier for syntax to be applied for content-based discovery
contentFilterQuery	0..1	content	The query string shall be specified when contentFilterSyntax parameter is present. See clause 7.3.3.17.13 for applicable syntax for content-based discovery
level	0..1	(not applicable)	Constraint on maximum number of levels in the resource tree that retrieve operation shall span, see clause 7.3.3.17.14
offset	0..1	(not applicable)	The number of direct child and descendant resources the Hosting CSE shall skip over and not include within a retrieve response, see clause 7.3.3.17.15
applyRelativePath	0..1	(not applicable)	A resource tree relative path (e.g. ../tempContainer/LATEST) which applies after all the matching conditions have been used (i.e. a matching result has been obtained). See clause 7.2.2.17.17

7.3.3.17.1 Conditions on the creationTime attribute

The ***Filter Criteria*** elements ***createdBefore*** and ***createdAfter*** define a time interval which is tested against the *creationTime* attribute of the applicable resources in order to determine the matching resources.

This filter criterion shall be satisfied if any of the following three conditions is fulfilled:

1) only ***createdBefore*** given in Filter Criteria:
creationTime < ***createdBefore***

2) only ***createdAfter*** given in Filter Criteria:
createdAfter ≤ *creationTime*)

3) both ***createdBefore*** and ***createdAfter*** given in Filter Criteria:
(***createdAfter*** ≤ *creationTime*) AND (*creationTime* < ***createdBefore***)

NOTE: In case 3) the ***Filter Criteria*** will only generate a match if ***createdAfter*** < ***createdBefore***.

7.3.3.17.2 Conditions on the lastModifiedTime attribute

The ***Filter Criteria*** elements ***lastModifiedBefore*** and ***lastModifiedAfter*** define a time interval which is tested against the *lastModifiedTime* attribute of the applicable resources in order to determine the matching resources.

This filter criterion shall be satisfied if any of the following three conditions is fulfilled:

1) only ***unmodifiedSince*** given in ***Filter Criteria***:
lastModifiedTime < ***unmodifiedSince***

2) only ***modifiedSince*** given in ***Filter Criteria***:
modifiedSince ≤ *lastModifiedTime*

3) both ***unmodifiedSince*** and ***modifiedSince*** given in ***Filter Criteria***:
(***modifiedSince*** ≤ *lastModifiedTime*) AND (*lastModifiedTime* < ***unmodifiedSince***))

NOTE: In case 3) the ***Filter Criteria*** will only generate a match if ***modifiedSince*** < ***unmodifiedSince***.

7.3.3.17.3 Conditions on stateTag attribute

The Filter Criteria elements ***stateTagSmaller*** and ***stateTagBigger*** define a number range which is tested against the stateTag attribute of the applicable resources in order to determine the matching resources.

This filter criterion shall be satisfied if any of the following three conditions is fulfilled:

1) only ***stateTagSmaller*** given in ***Filter Criteria***:
stateTag < ***stateTagSmaller***

2) only ***stateTagBigger*** given in ***Filter Criteria***:
stateTagBigger ≤ *stateTag*

3) both ***stateTagSmaller*** and ***stateTagBigger*** given in ***Filter Criteria***:
(***stateTagBigger*** ≤ *stateTag*) AND (*stateTag* < ***stateTagSmaller***)

NOTE: In case 3) the ***Filter Criteria*** will only generate a match if ***stateTagBigger < stateTagSmaller.***

7.3.3.17.4 Conditions on expirationTime attribute

The ***Filter Criteria*** elements ***expireBefore*** and ***expireAfter*** define a time interval which is tested against the *expirationTime* attribute of the applicable resources in order to determine the matching resources.

This filter criterion shall be satisfied if any of the following three conditions is fulfilled:

1) only ***expireBefore*** given in ***Filter Criteria***:
expirationTime < ***expireBefore***

2) only ***expireAfter*** given in ***Filter Criteria***:
expireAfter ≤ *expirationTime*

3) both ***expireBefore*** and ***expireAfter*** given in ***Filter Criteria***:
(***expireAfter*** ≤ *expirationTime*) AND (*expirationTime* < ***expireBefore***)

NOTE: In case 3) the Filter Criteria will only generate a match if ***expireAfter*** < ***expireBefore***.

7.3.3.17.5 Conditions on labels attribute

Each of the ***Filter Criteria*** elements ***labels, childLabels*** and ***parentLabels*** defines a list of labels which is tested against the *labels* attribute of applicable resource instances in order to determine the matching resources.

This filter criterion shall be satisfied if any of the ***labels*** in Filter Criteria matches any of the *labels* in the respective resource attribute. The matched resources are:

- the resources with the matched *label* attribute, when applying the ***Filter Criteria*** element ***labels;***

- the parent resources of the resources with the matched *label* attribute, when applying the ***Filter Criteria*** element ***childLabels;***
- the child resources of the resources with the matched *label* attribute, when applying the ***Filter Criteria*** element ***parentLabels.***

7.3.3.17.6 Conditions on resourceType attribute

Each of the ***Filter Criteria*** elements ***resourceType, childResourceType and parentResourceType*** defines a list of resource types which is tested against the *resourceType* attribute of the applicable resource instances in order to determine the matching resources.

This filter criterion shall be satisfied if any of the numbers in the ***resourceType Filter Criteria*** element matches the *resourceType* attribute. The matched resources are:

- the resources with the matched *resourceType* attribute, when applying the ***Filter Criteria*** element ***resourceType;***
- the parent resources of the resources with the matched *resourceType* attribute, when applying the ***Filter Criteria*** element ***childResourceType;***
- the child resources of the resources with the matched *resourceType* attribute, when applying the ***Filter Criteria*** element ***parentResourceType.***

7.3.3.17.7 Conditions on contentSize attribute

The ***Filter Criteria*** elements ***sizeBelow*** and ***sizeAbove*** define a number range which is tested against the value of the *contentSize* attribute of applicable <contentInstance> resources in order to determine the matching resources.

This filter criterion shall be satisfied if any of the following three conditions is fulfilled:

1) only ***sizeBelow*** given in ***Filter Criteria***:
   contentSize < ***sizeBelow***
2) only ***sizeAbove*** given in ***Filter Criteria***:
   sizeAbove ≤ *contentSize*
3) both ***sizeBelow*** and ***sizeAbove*** given in ***Filter Criteria***:
   (***sizeAbove*** ≤ *contentSize*) AND (*contentSize* < ***sizeBelow***)

NOTE: In case 3) the Filter Criteria will only generate a match if ***sizeAbove*** < ***sizeBelow***.

7.3.3.17.8 Conditions on typeOfContent of contentInfo attribute

The ***Filter Criteria*** element ***typeOfContent*** defines a string (or multiple such strings) which is compared against the *contentInfo* attribute of applicable <*contentInstance*> resources in order to determine the matching resources.

One or multiple such *typeOfContent* elements may be included in the ***Filter Criteria*** parameter.

This filter criterion shall be satisfied if any of the ***typeOfContent*** elements in ***Filter Criteria*** matches the *typeOfContent* part of the *contentInfo* attribute in a <*contentInstance*> resource.

7.3.3.17.9 Conditions on attribute name and value pairs

Each of the ***Filter Criteria*** elements ***attribute, childAttribute and parentAttribute*** defines one or more attribute name/value pairs (see clause 6.3.5.8) that are compared against all applicable resource representations which include a resource attribute with a name matching the attribute name in the ***Filter Criteria***, in order to determine the matching resources.

This filter criterion shall be satisfied if any of the attribute elements in the ***Filter Criteria*** parameter matches both attribute name and attribute value in any applicable resource representation.

The following attributes shall not be used for this condition as they may conflict with other ***Filter Criteria*** parameters: *creationTime*, *lastModifiedTime*, *stateTag*, *expirationTime*, *labels*, *resourceType*, *contentSize* and *contentInfo*.

The attribute value may include wildcard (*) in case of discovery requests. The wildcard (*) can match 0 or more characters. For example, a wildcard can be used within the value of a *creator* attribute condition (E.g.creator=CAE-ID* or creator=*ID123).

The matched resources are:

- the resources with the matched attribute elements, when applying the ***Filter Criteria*** element ***resourceAttribute;***
- the parent resources of the resources with the matched attribute elements, when applying the ***Filter Criteria*** element ***childAttribute;***
- the child resources of the resources with the matched attribute elements, when applying the ***Filter Criteria*** element ***parentAttribute.***

7.3.3.17.10 Constraint on number of retrieved resources by limit element

The ***limit*** element of the ***Filter Criteria*** parameter does not represent a matching condition. It imposes a limit on the number of resources that should be retrieved with the given Retrieve request primitive.

The number of resources retrieved with the request primitive (and to be included into the corresponding response primitive) shall not exceed the number indicated in the ***limit*** element of the ***Filter Criteria*** parameter.

7.3.3.17.11 Filter Usage request parameter

The ***filterUsage*** element of the ***Filter Criteria*** parameter does not represent a matching condition. It indicates how the ***Filter Criteria*** parameter shall be used. If this parameter is not provided, the Retrieve request primitive which includes this element triggers a generic retrieve operation. The data type of ***filterUsage*** is defined in clause 6.3.4.2.31.

7.3.3.17.12 Filter Operation request attribute

The ***filterOperation*** element of the ***Filter Criteria*** parameter does not represent a matching condition. It indicates the logical operation to be performed between the condition tags when multiple condition tags of different types are specified. If this parameter is not provided logical AND operation shall be used by default. The data type of ***filterOperation*** is defined in clause 6.3.4.2.34.

7.3.3.17.13 Conditions on content-based discovery

7.3.3.17.13.1 Introduction

The ***Filter Criteria*** elements ***contentFilterSyntax*** and ***contentFilterQuery*** are used for "content based discovery".

The ***contentFilterSyntax*** element specifies the syntax to be used by the ***contentFilterQuery*** element to express the content-based discovery conditions. A ***contentFilterQuery*** shall be included if the ***contentFilterSyntax*** parameter is present.

The ***contentFilterQuery*** element contains the query expression for content-based discovery.

7.3.3.17.13.2 JSON path syntax

When the ***contentFilterSyntax*** element has the value "JSON_PATH_SYNTAX", the ***contentFilterQuery*** element shall be expressed using "jsonpath query syntax" (see clause K.2 of oneM2M TS-0001 [6]). If the Hosting CSE does not support this syntax, NOT_IMPLEMENTED error shall be returned.

7.3.3.17.14 Constraint on number of applicable levels in resource tree

The ***level*** element of the ***Filter Criteria*** parameter does not represent a filter condition. It imposes a maximum limit on the depth of child resources in the resource tree that the Hosting CSE shall perform a Retrieve request upon. The ***level*** of 1 shall indicate the direct child resources.

7.3.3.17.15 Constraint on number of resources to skip over in retrieve response

The ***offset*** element of the ***Filter Criteria*** parameter does not represent a filter condition. It specifies the number of direct child resources and their descendants that the Hosting CSE shall skip over and not include within the response to a Retrieve request primitive. An ***offset*** of 1 shall indicate the first direct child resource.

7.3.3.17.16 Conditions on labelsQuery attribute

The ***Filter Criteria*** element ***labelsQuery*** contains a list of expressions to be tested against the *labels* attribute of the applicable resource instances.

This filter criterion shall be satisfied if any of the expressions in Filter Criteria matches the *labels* in the respective resource attribute.

Expressions can be in the following formats:

- key

Matches if *labels* attribute of the resource contains the key either in a key-only or key-value format.

For example: labels = color will match resources with *labels* defined as "color", "color:red" or "color:green".

- NT key

NT stands for not. Matches if *labels* attribute of the resource does not contain the key either in a key-only or key-value format.

For example: labels = NT color will match resources with labels that do not include "color" as a key.

- Key EQ value or key:value

EQ stands for equals. Matches if *labels* attribute of the resource contains the key-value pair.

For example: labels = color EQ red will match resources with *labels* that contains "color:red". Labels = color:red will match resources with *labels* defined as "color:red".

- Key NE value

NE stands for Not Equals to. Matches if the *labels* attribute of the resource contains at least one key-value pair that matches the key but none of these key-value pairs matches the value.

For example: labels = color NE red will match resources with labels that contains one key of "color" but with value not "red".

- Key IN (value1, value2…)

Matches if the *labels* attribute of the resource contains at least one key-value pair matching the key and one of the value listed.

For example: labels = color IN (red, green) will match resources with labels that contains "color:red" or "color:green"

- Key NI (value1, value2…)

NI stands for Not In. Matches if the *labels* attribute of the resource contains at least one key-value pair matching the key but none of these key-value pairs have any of the values listed.

For example: labels = color NI (red, green) will match resources with labels that contains "color:yellow" or "color:blue". Any resource that has label "color:red" or "color:green" will not be matched.

7.3.3.17.17 Applying a relative path

The ***applyRelativePath*** element of the ***Filter Criteria*** parameter does not represent a matching condition, it is a condition which applies after all the matching conditions have been used (i.e. a matching result has been obtained) in order to determine the set of resources in the final filtering result. The filtering result is computed by appending the relative path to each of the URIs of the resources in the matching result. All resources that exist at the resulting path(s)

shall form the filtering result. If the combined relative path does not represent a valid resource, the outcome is the same as if no match was found, i.e. there is no corresponding entry in the filtering result.

7.3.3.18 Semantic resource discovery

7.3.3.18.0 Introduction

Semantic resource discovery is used to find resources in a CSE based on the semantic descriptions contained in the *descriptor* attribute of <*semanticDescriptor*> resources. Since an overall semantic description (forming a graph [i.5]) may be distributed across a set of <semanticDescriptor> resources, the semantic descriptions have to be retrieved (before or as needed) during the execution of the discovery request.

Semantic resource discovery is initiated by sending a Retrieve request with the discovery criteria in the ***semanticsFilter*** filter condition(s) with two alternatives:

a) Targeting a <*semanticFanOutPoint*> virtual resource, see clause 7.4.35.

b) Targeting a resource other than <semanticFanOutPoint>. In this alternative the semantic resource discovery request procedure shall be comprised of the following actions:

Originator:

The Originator shall follow the steps from Orig-1.0 to Orig-6.0 specified in clause 7.2.2.1 Generic Resource Request Procedure for Originator.

In addition to Orig-1.0, the following steps shall be performed.

The *To* parameter in the Retrieve Request shall indicate the root of where the semantic discovery begins.

The *filterCriteria* of the Retrieve Request shall include the *filterUsage* parameter configured as "discovery" and the ***semanticsFilter*** filter condition.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-7.0 specified in clause 7.2.2.2 Generic Resource Request Procedure for Receiver.

After Recv-1.0 "Check the validity of received request primitive": check that the syntax of the ***semanticsFilter*** corresponds to a valid SPARQL query request [33]. If the ***semanticsFilter*** content does not correspond to a valid SPARQL query request, the Receiver shall generate a Response Status Code indicating a "NOT_ACCEPTABLE" error.

The Hosting CSE shall follow the steps from Recv-1.0 to Recv-6.2 specified in clause 7.2.2.2.The Hosting CSE shall not perform steps from Recv-6.3 to Recv-6.6 and perform the following steps instead:

1) The Hosting CSE shall find the <semanticDescriptor> resource(s) to which the Originator has "Discover" access right, under the addressed resource.

   a) If the *relatedSemantics* attribute does not exist, the "Annotation-based method" (using *resourceDescriptorLink)* detailed in clause 7.3.3.18.1 shall be used.

   b) If the *relatedSemantics* attribute exists the "Resource link-based method" (using the *relatedSemantics attribute*) detailed in clause 7.3.3.18.2 shall be used.

2) The Hosting CSE shall perform Recv-6.7 "Create a success response" where the Response shall include the resources matched based on the SPARQL engine result.

7.3.3.18.1 Annotation-based method

In the annotation-based method, related <semanticDescriptor> resources are identified within the RDF semantic description itself using a special annotation property called *m2m:resourceDescriptorLink*. This property points to another <semanticDescriptor> resource which may contain relevant information for matching the semantic filter. Whenever, during the execution of the SPARQL request (on the semantic description in the *descriptor* attribute of the <semanticDescriptor>) such an annotation property is found, the execution is halted, the content of the *descriptor*

attribute of the referred to <semanticDescriptor> is retrieved, and the execution is continued on the combined content of the already present and the just retrieved semantic information.

7.3.3.18.2 Resource link-based method

In this option, the *relatedSemantics* attribute contains the list of <semanticDescriptor> resources which shall be retrieved for the purpose of creating the overall graph against which the SPARQL request is executed.

The Hosting CSE retrieves the <semanticDescriptor> child resource of the request target and the addresses provided in the *relatedSemantics* attribute. For each address from the *relatedSemantics* list the Hosting CSE:

- checks that the Originator has "Discover" access rights, and the existence of the addressed resource;
- retrieves the description in the *descriptor* attribute under the addressed resource.

The Hosting CSE shall aggregate all the retrieved descriptors and deliver the content for SPARQL request processing, along with the ***semanticsFilter*** content.

NOTE: In the resource link-based method, no actions need to be performed during the execution of the SPARQL request if the notation property *onem2m:resourceDescriptorLink* is encountered.

Afterwards, the Hosting CSE performs Recv-6.7 "Create a success response" where the Response shall include the resources matched based on the SPARQL engine result.

7.3.3.19 Semantic query

7.3.3.19.0 Introduction

Semantic queries enable the retrieval of both explicitly and implicitly derived information based on syntactic, semantic and structural information contained in data (such as RDF data). The result of a semantic query is the semantic information/knowledge for answering/matching the query. Note that, in the following descriptions, the general term semantic resource is used to refer to <semanticDescriptor> resources and any other future resources containing semantic information.

For a given semantic query, it needs to be executed on a set of RDF triples (called the "RDF data basis"), which may be distributed in the resource tree and stored in different semantic resources. The Receiver shall perform semantic graph scoping, which is the process of establishing the "query scope" for this semantic query in order to build its RDF data basis. The following two approaches may be used to decide the semantic query scope of a semantic query:

Approach-1: The scope of the semantic query is provided implicitly.

Approach-2: The scope of the semantic query is provided explicitly

7.3.3.19.1 Approach-1: Semantic query with implicit scope

In Approach-1, a semantic query request message targets any resource (i.e. as specified by the ***To*** parameter) and the semantic query shall be executed relative to this target resource, similarly to other request messages. The scope of the semantic query is formed through the aggregation of the semantic contents of the target resource's descendants. All the contents of semantic resource descendants of the target resource shall form the RDF data basis for this semantic query to be executed on. In this alternative, the semantic query procedure shall be comprised of the following actions:

Originator:

The Originator shall follow the steps from Orig-1.0 to Orig-6.0 specified in clause 7.2.2.1 Generic Resource Request Procedure for Originator.

In addition to Orig-1.0, the following steps shall be performed.

The ***To*** parameter in the Retrieve Request shall define the scope of this semantic query as mentioned earlier.

The Retrieve Request shall include the following parameters:

1) the ***Semantic Query Indicator***, which is set to true;

2) ***filterCriteria*** of the Retrieve Request shall include the ***semanticsFilter*** condition tag; and

3) the parameter ***Result Content*** shall be set to "semantic content" to indicate that the response message shall contain the result of a semantic query.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-7.0 specified in clause 7.2.2.2 Generic Resource Request Procedure for Receiver.

After Recv-1.0 "Check the validity of received request primitive": check that the syntax of the ***semanticsFilter*** corresponds to a valid SPARQL query request [33]. If the ***semanticsFilter*** content does not correspond to a valid SPARQL query request, the Receiver shall generate a Response Status Code indicating a "NOT_ACCEPTABLE" error.

The Hosting CSE shall follow the steps from Recv-1.0 to Recv-6.2 specified in clause 7.2.2.2. The Hosting CSE shall not perform steps from Recv-6.3 to Recv-6.6 and perform the following steps instead:

1) The Hosting CSE shall find the semantic resources to which the Originator has "RETRIEVE" access right, under the addressed resource as specified by the ***To*** parameter.

2) Aggregate the semantic resources and deliver the content for SPARQL processing, along with the ***semanticsFilter*** content.

3) Wait for a SPARQL processing response.

4) Perform Recv-6.7 "Create a success response" where the Response shall include the SPARQL processing result, which is the semantic query result to be returned.

5) Perform Recv-6.8 and the procedure is terminated.

7.3.3.19.2 Approach-2: Semantic query with explicit scope

In Approach-2, the relevant semantic resources are the members of a <*group*> resource. The scope of the semantic query is formed through the aggregation of the semantic contents of all the group members. In this approach, the request targets the <semanticFanOutPoint> (as specified by the ***To*** parameter), i.e. the child resources of the <group> resource. As a result, this <group> resource explicitly specifies the RDF data basis of the semantic query. The details of Approach-2 are introduced in clause 7.4.35.

7.3.4 Management common operations

7.3.4.1 Identify the managed entity and the technology specific protocol

Where a managed entity is being addressed via a <mgmtObj> resource, the Hosting CSE shall identify the managed entity via the <node> resource that is the parent resource of the <mgmtObj> resource. In case of a <mgmtCmd> resource the entity to be managed is indicated by its *execTarget* attribute. This addresses either a <node> resource or a group of resources of type <node>. Hence, in all cases the managed entity is ultimately identified through a <node> resource, from which the identifier of the device can be retrieved.

The Hosting CSE shall determine the technology specific protocol to be used for communicating with the managed entity based on the *objectIDs* attribute of the addressed <mgmtObj> resource.

If the managed entity cannot be identified, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_REACHABLE" in the Response primitive.

7.3.4.2 Locate the technology specific data model objects to be managed on the managed entity

The Hosting CSE shall locate the technology specific data model object to be managed on the managed entity by the *objectPaths* attribute of the <mgmtObj> resource addressed by the URI provided in the ***To*** primitive parameter. In the case that the ***To*** addresses an [objectAttribute], the Hosting CSE shall locate the technology specific data model object on the managed entity through the *objectPaths* attribute of the <mgmtObj> resource of the addressed [objectAttribute], combined with their relative position in the technology specific data model object tree. If the technology specific data

model object cannot be located, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_FOUND" in the Response primitive.

In the case that the management server is external to the Hosting CSE, the Hosting CSE shall identify the management server that is capable of performing the operation on the technology specific data model object. If the management server cannot be identified, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_REACHABLE" in the Response primitive.

7.3.4.3 Establish a management session with the managed entity or management server

In the case that the management server is embedded with the CSE, if there is no existing management session between the Hosting CSE and the managed entity, the Hosting CSE shall also trigger the managed entity to establish a management session with the Hosting CSE by sending triggering message to the managed entity using the determined technology specific protocol in case such triggering mechanism is supported by the technology specific protocol. If the triggering mechanism is not supported by the technology specific protocol, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "MGMT_SESSION_CANNOT_BE_ESTABLISHED". If the management session cannot be established with the managed entity, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "MGMT_SESSION_CANNOT_BE_ESTABLISHED". If the management session cannot be established within a limited time span as per local policy, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "MGMT_SESSION_ESTABLISHMENT_TIMEOUT" in the Response primitive.

In the case that the management server is external to the Hosting CSE, if there is no existing management session between the Hosting CSE and the management server that manages the technology specific data model objects, the Hosting CSE shall establish a session with the managed entity with the necessary access control privileges to perform the technology specific request on the technology specific protocol. If the management session cannot be established with the management server, the Hosting CSE shall reject the request with ***Response Status Code*** indicating "MGMT_SESSION_CANNOT_BE_ESTABLISHED". If the management session cannot be established within a limited time span as per local policy, the Hosting CSE shall reject the request with ***Response Status Code*** indicating "MGMT_SESSION_ESTABLISHMENT_TIMEOUT" in the Response primitive.

7.3.4.4 Send the management request(s) to the managed entity corresponding to the received Request primitive

The Hosting CSE shall send the management request(s) to the managed entity or management server in the established management session in order to perform the management operation as requested by the received Request primitive. The management request shall address the technology-specific data model object on the managed entity as determined in clause 7.3.4 or in the primitive-specific clauses. The management request being used is specific to the technology specific protocol according to a pre-defined mapping relationship with the Request primitive. The internal data structure of the technology specific data model object addressed by the technology specific request shall be determined based on the mapping relationship of the <mgmtObj> or <mgmtCmd> resources and the technology specific data model objects or based on the generic mapping rule as specified in oneM2M TS-0001 [6], clauses 9.6.15, 9.6.16 and 9.6.17. The Hosting CSE shall extract the management results received from the managed entity or management server in order to prepare a Response primitive to be sent to the originator later. Unless explicitly stated, if the management request cannot be performed successfully, the Hosting CSE shall reject the Request primitive with the management server in the Response primitive according to the mapping relationship with the technology specific protocol.

7.4 Resource type-specific procedures and definitions

7.4.0 Introduction

This clause specifies the structure of each individual resource type and the resource type specific details of procedures (i.e. differences from the generic procedures specified in clauses 7.2 and 7.3) to be performed by the originator and receiver of a request message.

The applicability of each of the following resource type-specific procedures to an interface reference point (i.e. Mca, Mcc and Mcc') is defined in clause 10.2 (Resource Type-Specific Procedures) of oneM2M TS-0001 [6].

7.4.1 Resource type specification conventions

7.4.1.1 Resource type definition conventions

This clause provides general information on conventions applied to resource type specifications and how to interpret the provided information. Each resource type is defined in a tabular format as shown in the tables below. A reference to the XSD file associated with the given resource type is provided in the format shown in Table 7.4.1.1-1. Further information on usage and limitations of the XSD is given in clause 6.1.

Table 7.4.1.1-1: Data type definition of <resourceType>

Data Type ID	File Name	Note
Actual Data Type ID	XSD file name	

Information about universal/common resource attributes of a resource type is provided in the format shown in Table 7.4.1.1-2. The column "Request optionality" specifies the presence of each resource attribute in the ***Content*** parameter of the request primitive. This is defined as Mandatory (M), Optional (O), or NP (Not Present). The table applies only to Create and Update operations. A Retrieve request operation may include a ***Content*** parameter containing a list of attribute names to be retrieved. A Delete request shall not include a ***Content*** parameter.

Universal/common resource attributes do not have any default value. However, value restrictions and notes given in Table 6.3.6-1 apply.

Table 7.4.1.1-2: Universal/Common Attributes of <resourceType> resource

Attribute Name	Request Optionality	
	Create	Update
Universal/common attribute name	M/O/NP	O/NP

Information about resource specific attributes of the resource type is provided in the format shown in Table 7.4.1.1-3. Request optionality shall be interpreted the same way as described for universal/common attributes above.

Table 7.4.1.1-3: Resource Specific Attributes of <resourceType> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
resource specific attribute name	M/O/NP	O/NP	Assigned XSD datatype	Applicable constraints on attribute values and default value if applicable

Request primitives shall comply with the request optionality of universal/common and resource specific attributes in Create and Update request primitives as defined for each individual resource type in the following clauses. The values of each resource attribute shall comply with its assigned datatype and any indicated constraints. See clause 6.1 for limitations on using the XSD files associated with the resource type for validation of compliance.

Table 7.4.1.1-4 provides the information of child resources of the resource type.

Table 7.4.1.1-4: Child resources of <resourceType> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<resourceType>	"[variable]" or fixed name	Multiplicity of child resource instances in the corresponding resource type	Reference to the resource type definition in the present document.

7.4.1.2 Resource type-specific procedure conventions

This clause describes resource type specific procedures referring to generic procedures defined in clause 7.2.2. Each operation specific procedure describes procedures for the Originator and the Receiver. If the resource and operation specific procedure is the same as the generic procedure, the Originator and Receiver procedure refer to them. Otherwise, the deviation/addition is clearly described with related procedure numbers (e.g. Recv 6.1) in clause 7.2.2.

If a deviation/addition procedure includes sub-procedures in one more level(s), proper numbering is used to show the levels (e.g. "1)", "a)"). If sub-procedures do not care about the order, bullets are used instead of numbers.

7.4.2 Resource type <accessControlPolicy>

7.4.2.1 Introduction

The <accessControlPolicy> resource is comprised of *privileges* and *selfPrivileges* attributes which represent a set of access control rules defining which entities (defined as accessControlOriginators) have the privilege to perform certain operations (defined as accessControlOperations) within specified contexts (defined as accessControlContexts) and are used by the CSEs in making access decision to specific resources.

The detailed description can be found in clause 9.6.2 in oneM2M TS-0001 [6].

Table 7.4.2.1-1: Data type definition of <accessControlPolicy> resource

Data Type ID	File Name	Note
accessControlPolicy	CDT-accessControlPolicy-v3_11_0.xsd	

Table 7.4.2.1-2: Universal/Common Attributes of <accessControlPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
labels	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
announceTo	O	O
announcedAttribute	O	O

Table 7.4.2.1-3: Resource Specific Attributes of <accessControlPolicy> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
privileges	M	O	m2m:setOfAcrs	No default
selfPrivileges	M	O	m2m:setOfAcrs	No default At least one accessControlRule shall be present.
authorizationDecisionResourceIDs	O	O	m2m:listOfURIs	No default
authorizationPolicyResourceIDs	O	O	m2m:listOfURIs	No default
authorizationInformationResourceIDs	O	O	m2m:listOfURIs	No default

Table 7.4.2.1-4 includes information about the child resources of the resource type.

Table 7.4.2.1-4: Child Resources of <accessControlPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.2.2 accessControlPolicy resource specific procedures for CRUD operations

7.4.2.2.0 Introduction

This clause describes accessControlPolicy resource-specific behaviour for CRUD operations.

7.4.2.2.1 Create

Originator:

No changes from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the <accessControlPolicy> received does not have at least one accessControlRule specified in the *selfPrivileges* attribute then "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "BAD_REQUEST" error.

7.4.2.2.2 Retrieve

Originator:

No changes from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.2.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the <accessControlPolicy> received removes all accessControlRules specified in the *selfPrivileges* attribute then "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "BAD_REQUEST" error.

7.4.2.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.3 Resource Type <CSEBase>

7.4.3.1 Introduction

A <CSEBase> resource shall represent a CSE. This <CSEBase> resource shall be the root for all the resources that are residing on the CSE. The detailed description can be found in clause 9.6.3 in oneM2M TS-0001 [6].

Table 7.4.3.1-1: Data type definition of <CSEBase> resource

Data Type ID	File Name	Note
CSEBase	CDT-CSEBase-v3_11_0.xsd	

Table 7.4.3.1-2: Universal/Common Attributes of <CSEBase> resource

Attribute Name
@resourceName
resourceType
resourceID
parentID
creationTime
lastModifiedTime
labels

The value of the parentID attribute for the <CSEBase> resource shall be an empty string since the <CSEBase> resource does not have a parent. The common attributes *accessControlPolicyIDs* and *dynamicAuthorizationConsultationIDs* are treated as resource-specific attributes.

Table 7.4.3.1-3: Resource Specific Attributes of <CSEBase> resource

Attribute Name	Data Type	Default Value and Constraints
accessControlPolicyIDs	m2m:acpType	No default
cseType	m2m:cseTypeID	No default
CSE-ID	m2m:ID	No default
supportedResourceType	list of m2m:resourceType	No default
pointOfAccess	m2m:poaList	No default
nodeLink	xs:anyURI	No default
dynamicAuthorizationConsultationIDs	list of xs:anyURI	No default
contentSerialization	m2m:serializations	No default
e2eSecInfo	m2m:e2eSecInfo	No default
supportedReleaseVersions	m2m:supportedReleaseVersions	No default

Table 7.4.3.1-4: Child resources of <CSEBase> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<remoteCSE>	[variable]	0..n	Clause 7.4.4
<remoteCSEAnnc>	[variable]	0..n	Clause 7.4.4
<node>	[variable]	0..n	Clause 7.4.18
<AE>	[variable]	0..n	Clause 7.4.5
<container>	[variable]	0..n	Clause 7.4.6
<group>	[variable]	0..n	Clause 7.4.13
<accessControlPolicy>	[variable]	0..n	Clause 7.4.2
<subscription>	[variable]	0..n	Clause 7.4.8
<mgmtCmd>	[variable]	0..n	Clause 7.4.16
<locationPolicy>	[variable]	0..n	Clause 7.4.10
<statsConfig>	[variable]	0..n	Clause 7.4.23
<statsCollect>	[variable]	0..n	Clause 7.4.25
<request>	[variable]	0..n	Clause 7.4.12
<delivery>	[variable]	0..n	Clause 7.4.11
<schedule>	[variable]	0..1	Clause 7.4.9
<m2mServiceSubscriptionProfile>	[variable]	0..n	Clause 7.4.19
<serviceSubscribedAppRule>	[variable]	0..n	Clause 7.4.29
<notificationTargetPolicy>	[variable]	0..n	Clause 7.4.31
<dynamicAuthorizationConsultation>	[variable]	0..n	Clause 7.4.36
<flexContainer>	[variable]	0..n	Clause 7.4.37
<timeSeries>	[variable]	0..n	Clause 7.4.38
<role>	[variable]	0..n	Clause 7.4.40
<token>	[variable]	0..n	Clause 7.4.41
<authorizationDecision>	[variable]	0..n	Clause 7.4.43
<authorizationPolicy>	[variable]	0..n	Clause 7.4.44
<authorizationInformation>	[variable]	0..n	Clause 7.4.45
<ontologyRepository>	[variable]	0..1	Clause 7.4.46
<semanticMashupJobProfile>	[variable]	0..n	Clause 7.4.49
<semanticMashupInstance>	[variable]	0..n	Clause 7.4.50
<AEContactList>	[variable]	0..1	Clause 7.4.53
<localMulticastGroup>	[variable]	0..n	Clause 7.4.55
<crossResourceSubscription>	[variable]	0..n	Clause 7.4.58
<backgroundDataTransfer>	[variable]	0..n	Clause 7.4.59
<transactionMgmt>	[variable]	0..n	Clause 7.4.60
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.3.2 <CSEBase> resource specific procedures for CRUD+N operations

7.4.3.2.1 Create

Originator:

The <CSEBase> resource shall not be created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order.

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.3.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.3.2.3 Update

Originator:

The <CSEBase> resource shall not be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order.

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.3.2.4 Delete

Originator:

The <CSEBase> resource shall not be deleted via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order.

   a) "Create an unsuccessful Response primitive" with the Response Status Code indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.3.2.5 Notify

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.4 Resource Type <remoteCSE>

7.4.4.1 Introduction

A <remoteCSE> resource shall represent a remote CSE that is registered to the Registrar CSE. <remoteCSE> resources shall be located directly under the <CSEBase> of the Registrar CSE.

In addition each registered CSE shall have a <remoteCSE> resource representing its Registrar CSE. This is located directly under the registered CSE's <CSEBase>.

The detailed description can be found in clause 9.6.4 in oneM2M TS-0001 [6].

Table 7.4.4.1-1: Data type definition of <remoteCSE> resource

Data Type ID	File Name	Note
remoteCSE	CDT-remoteCSE-v3_11_0.xsd	

Table 7.4.4.1-2: Universal/Common Attributes of <remoteCSE> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.4.1-3: Resource Specific Attributes of <remoteCSE> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
cseType	O	NP	m2m:cseTypeID	No default
pointOfAccess	O	O	m2m:poaList	No default
CSEBase	M	NP	xs:anyURI	No default
CSE-ID	M	NP	m2m:ID	No default
M2M-Ext-ID	O	O	m2m:externalID	No default
Trigger-Recipient-ID	O	O	m2m:triggerRecipientID	No default
requestReachability	M	O	xs:boolean	No default
nodeLink	O	O	xs:anyURI	No default
triggerReferenceNumber	O	O	xs:unsignedInt	No default
contentSerialization	O	O	m2m:serializations	No default
e2eSecInfo	O	O	m2m:e2eSecInfo	No default
descendantCSEs	O	O	m2m:listOfM2MID	No default
supportedReleaseVersions	M	O	m2m:supportedReleaseVersions	No default
multicastCapability	O	O	m2m:multicastCapability	No default
externalGroupID	O	O	m2m:externalID	No default
triggerEnable	O	O	xs:boolean	false
activityPatternElements	O	O	m2m:activityPatternElements	No default

Table 7.4.4.1-4: Child resources of <remoteCSE> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<container>	[variable]	0..n	Clause 7.4.6
<containerAnnc>	[variable]	0..n	Clause 7.4.6
<flexContainer>	[variable]	0..n	Clause 7.4.37
<flexContainerAnnc>	[variable]	0..n	Clause 7.4.37
<group>	[variable]	0..n	Clause 7.4.13
<groupAnnc>	[variable]	0..n	Clause 7.4.13
<accessControlPolicy>	[variable]	0..n	Clause 7.4.2
<accessControlPolicyAnnc>	[variable]	0..n	Clause 7.4.2
<subscription>	[variable]	0..n	Clause 7.4.8
<pollingChannel>	[variable]	0..1	Clause 7.4.21
<nodeAnnc>	[variable]	0..n	Clause 7.4.36
<dynamicAuthorizationConsultation>	[variable]	0..n	Clause 7.4.36
<flexContainer>	[variable]	0..n	Clause 7.4.37
<timeSeries>	[variable]	0..n	Clause 7.4.38
<timeSeriesAnnc>	[variable]	0..n	Clause 7.4.38
<remoteCSEAnnc>	[variable]	0..n	Clause 7.4.4
<AEAnnc>	[variable]	0..n	Clause 7.4.5
<locationPolicyAnnc>	[variable]	0..n	Clause 7.4.10
<ontologyRepositoryAnnc>	[variable]	0..1	Clause 7.4.46
<semanticMashupJobProfile>	[variable]	0..n	Clause 7.4.49
<semanticMashupJobProfileAnnc>	[variable]	0..n	Clause 7.4.49
<semanticMashupInstance>	[variable]	0..n	Clause 7.4.50
<semanticMashupInstanceAnnc>	[variable]	0..n	Clause 7.4.50
<crossResourceSubscription>	[variable]	0..n	Clause 7.4.58
<transactionMgmt>	[variable]	0..n	Clause 7.4.60
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.4.2 <remoteCSE> resource specific procedures for CRUD operations

7.4.4.2.0 Introduction

The entire CSE registration procedure including <remoteCSE> resource creation procedure below is defined in clause 10.1.1.2.1 in oneM2M TS-0001 [6] ("CSE registration procedure").

7.4.4.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

- An AE shall not originate a Create <remoteCSE> resource request.
- The Originator, upon receipt of successful CREATE response message, shall create a <remoteCSE> resource locally. It shall populate this resource with all the mandatory attributes for a <remoteCSE>.
- The Originator may issue a RETRIEVE request to the registrar <CSEBase> to fetch additional information about the registrar CSE and add this information to its local <remoteCSE> resource. The Originator may choose which optional attributes to include in its <remoteCSE> resource.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except replacement of Recv-6.3 step with following:

1) The Hosting CSE shall check if the credential provided by the Originator is valid.
2) If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "SECURITY_ASSOCIATION_REQUIRED" error.

And addition of the following to step Recv-6.4:

1) The Hosting CSE shall check for the presence of any resources having a CSE-ID that matches the one specified in the request and that have the same parent as the new resource being created.

2) If such a resource exists, then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "ORIGINATOR_HAS_ALREADY_REGISTERED" error.

And addition of the following to step Recv-6.5:

1) If the Receiver CSE has registered to another CSE, the Receiver CSE shall send an update request to its Registrar CSE to add the CSE-IDs of the Originator CSE and the Originator CSE's descendants into the *descendantCSEs* attribute of the Receiver CSE's <remoteCSE> hosted by the Registrar CSE.

7.4.4.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.4.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

- An AE shall not originate a Update <remoteCSE> resource request.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except addition of the following to step Recv-6.5:

If the *descendantCSEs* attribute is updated, and the Receiver CSE has registered to another CSE, the Receiver CSE shall send an update request to its Registrar CSE to make the corresponding updates to the *descendantCSEs* attribute of the Receiver CSE's <remoteCSE> hosted by the Registrar CSE.

7.4.4.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exceptions:

- An AE shall not originate a Delete <remoteCSE> resource request.
- The Originator, upon receipt of successful DELETE response message, shall delete the corresponding <remoteCSE> resource locally.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except addition of the following to step Recv-6.5:

If the Receiver CSE has registered to another CSE, the Receiver CSE shall send an update request to its Registrar CSE to delete the CSE-IDs of the Originator CSE and the Originator CSE's descendants from the *descendantCSEs* attribute of the Receiver CSE's <remoteCSE> hosted by the Registrar CSE.

7.4.5 Resource Type <AE>

7.4.5.1 Introduction

The <AE> resource represents information about an Application Entity known to a given Common Services Entity.

The detailed description can be found in clause 9.6.5 in oneM2M TS-0001 [6].

Table 7.4.5.1-1: Data type definition of <AE> resource

Data Type ID	File Name	Note
AE	CDT-AE-v3_11_0.xsd	XSD schema for AE resource

Table 7.4.5.1-2: Universal/Common Attributes of <AE> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.5.1-3: Resource Specific Attributes of <AE> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
appName	O	O	xs:string	No default
App-ID	M	NP	xs:string	No default
AE-ID	NP	NP	m2m:ID	No default
pointOfAccess	O	O	m2m:poaList	No default
ontologyRef	O	O	xs:anyURI	No default
nodeLink	O	O	xs:anyURI	No default
requestReachability	M	O	xs:boolean	No default
contentSerialization	O	O	m2m:serializations	No default
e2eSecInfo	O	O	m2m:e2eSecInfo	No default
M2M-Ext-ID	O	O	m2m:externalID	No default
supportedReleaseVersions	M	O	m2m:supportedReleaseVersions	No default
registrationStatus	O	O	m2m:AERegistrationStatus	No default
trackRegistrationPoints	O	O	xs:boolean	No default
sessionCapabilities	O	O	m2m:sessionCapabilities	No default
triggerEnable	O	O	xs:boolean	false
activityPatternElements	O	O	m2m:activityPatternElements	No default

Table 7.4.5.1-4: Child resources of <AE> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<container>	[variable]	0..n	Clause 7.4.6
<group>	[variable]	0..n	Clause 7.4.13
<accessControlPolicy>	[variable]	0..n	Clause 7.4.2
<pollingChannel>	[variable]	0..n	Clause 7.4.21
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<dynamicAuthorizationConsultation>	[variable]	0..n	Clause 7.4.36
<flexContainer>	[variable]	0..n	Clause 7.4.37
<timeSeries>	[variable]	0..n	Clause 7.4.38
<semanticMashupInstance>	[variable]	0..n	Clause 7.4.50
<multimediaSession>	[variable]	0..n	Clause 7.4.56
<triggerRequest>	[variable]	0..n	Clause 7.4.57
<crossResourceSubscription>	[variable]	0..n	Clause 7.4.58
<transactionMgmt>	[variable]	0..n	Clause 7.4.60
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.5.2 <AE> resource specific procedures for CRUD+N operations

7.4.5.2.0 Introduction

This clause describes AE resource specific behaviour for CRUD+N operations.

The entire AE registration procedure including <AE> resource creation procedure below is defined in clause 10.1.1.2.2 of oneM2M TS-0001 [6] ("Application Entity registration procedure").

7.4.5.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1 with the with the following exception:

- A CSE shall not originate a Create <AE> resource request.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except replacement of Recv-6.3 procedure as follows:

1) The <serviceSubscribedNode> resource associated with the Hosting CSE shall be checked, using the credential provided by the Originator, to determine if the requested registration is allowed by any of <serviceSubscribedAppRule> resources which are linked from the *ruleLinks* attribute.

2) If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "SECURITY_ASSOCIATION_REQUIRED" error.

Additional primitive specific operations on Recv-6.0.1 "Requested operation is an AE registration?":

1) If the request is for an <AE> resource and if it is received at a MN-CSE or ASN-CSE with "/S" in the ***From*** parameter, but no specific AE-ID-Stem was provided with the CREATE request of the Registree AE, then this is an initial registration and the receiver shall execute the following steps in order:

   a) Compose a Create <AEAnnc> Request primitive with the following attribute values:

      - The *link* attribute of the <AEAnnc> resource to be created shall be set to the SP-Relative-Resource-ID format of a - not yet existent - <*AE*> resource hosted on the Registrar CSE constructed with a Unstructured-CSE-relative-Resource-ID that is equal to the AE-ID-Stem value used for the Registree AE.

      - The *App-ID* attribute of the <*AEAnnc*> resource to be created shall be set to the *App-ID* attribute value of the Registree AE.

- The concatenation of the string 'Credential-ID:' and the actual Credential-ID of the Security Association used by the Registree AE - if any - shall be placed into the *labels* attribute of the <*AEAnnc*> resource. If no Security Association was used by the Registree AE, a value of 'None' shall be used for Credential-ID.

b) The ***From*** parameter of the CREATE request for the <*AEAnnc*> resource shall be set to the SP-relative-CSE-ID or Absolute-CSE-ID followed by '/S'.

c) Send Create request to the IN-CSE that is associated with the Registree AE.

d) Wait for Response primitive.

- Upon receipt of a successful response from the IN-CSE, the Registrar/Host CSE shall use the Unstructured-CSE-relative-Resource-ID that was used for the <*AEAnnc*> resource on the IN-CSE also as the assigned Unstructured-CSE-relative-Resource-ID for the <*AE*> resource to be created on the Registrar/Host CSE.

2) If the request is for an <AE> resource and if it is received at a MN-CSE or ASN-CSE with an AE-ID-Stem starting with "S", and this is an initial registration or a re-registration to the same Registrar CSE, then the receiver shall execute the following steps in order:

a) Determine if <AEAnnc> resource already exists in the IN-CSE for the Registree AE. If so, compose an Update <AEAnnc> Request primitive to update the <AEAnnc> resource. If no <AEAnnc> resource already exists in the IN-CSE for the Registree AE, the receiver shall compose a Create <AEAnnc> Request primitive. In both cases, the request primitive shall have the following attribute values:

- The *link* attribute of the <AEAnnc> resource shall be set or updated to the SP-Relative-Resource-ID format of a - not yet existent - <AE> resource hosted on the Registrar CSE constructed with a Unstructured-CSE-relative-Resource-ID that is equal to the AE-ID-Stem value used for the Registree AE.

- The *labels* attribute of the <AEAnnc> resource shall be set or updated to the concatenation of the string 'Credential-ID:' and the Credential-ID of the Security Association used by the Registree AE, replacing the existing entry starting with 'Credential-ID:' if present. If no Security Association was used by the Registree AE, a value of 'None' shall be used for Credential-ID.

b) The ***From*** parameter of the CREATE or UPDATE request for the <*AEAnnc*> resource shall be set to the SP-relative-CSE-ID or Absolute-CSE-ID followed by '/' and the AE-ID-Stem value.

c) The ***To*** parameter shall contain the SP-relative-Resource-ID format of the Resource ID for the <*AEAnnc*> resource which shall be constructed from the CSE-ID of the IN-CSE and the AE-ID-Stem that the Registree AE provided.

d) Send Create or Update request to the IN-CSE that is associated with the Registree AE.

e) Wait for Response primitive.

- Upon receipt of a successful response from the IN-CSE, the Registrar/Host CSE shall use the Unstructured-CSE-relative-Resource-ID equal to the AE-ID-Stem provided by the Registree AE for the <AE> resource to be created on the Registrar/Host CSE.

3) If the request is for an <AE> resource and if it is received at a MN-CSE or ASN-CSE with an AE-ID-Stem starting with "S", and this is a re-registration to a new Registrar CSE, then the receiver shall execute the following steps in order:

a) Determine if <AEAnnc> resource already exists in the IN-CSE for the Registree AE. If so, compose an Update <AEAnnc> Request primitive to update the <AEAnnc> resource. If no <AEAnnc> resource already exists in the IN-CSE for the Registree AE, the receiver shall compose a Create <AEAnnc> Request primitive. In both cases, the request primitive shall have the following attribute values:

- The *link* attribute of the <AEAnnc> resource shall be set or updated to the SP-Relative-Resource-ID format of a - not yet existent - <*AE*> resource hosted on the Registrar CSE constructed with an Unstructured-CSE-relative-Resource-ID that is equal to the AE-ID-Stem value used for the Registree AE.

- The *labels* attribute of the <AEAnnc> resource shall be set or updated to the concatenation of the string 'Credential-ID:' and the Credential-ID of the Security Association used by the Registree AE, replacing the existing entry starting with 'Credential-ID:' if present. If no Security Association was used by the Registree AE, a value of 'None' shall be used for Credential-ID.

b) The ***From*** parameter of the CREATE or UPDATE request for the *<AEAnnc>* resource shall be set to the SP-relative-CSE-ID or Absolute-CSE-ID followed by '/' and the AE-ID-Stem value.

c) The ***To*** parameter shall contain the SP-relative-Resource-ID format of the Resource ID for the *<AEAnnc>* resource which shall be constructed from the CSE-ID of the IN-CSE and the AE-ID-Stem that the Registree AE provided.

d) Send Create or Update request to the IN-CSE that is associated with the Registree AE.

e) Wait for Response primitive.

- Upon receipt of a successful response from the IN-CSE, the Registrar/Host CSE shall use the Unstructured-CSE-relative-Resource-ID equal to the AE-ID-Stem provided by the Registree AE for the <AE> resource to be created on the Registrar/Host CSE.

And addition of the following to step Recv-6.4:

- The Hosting CSE shall check for the presence of any resources having a AE-ID that matches the one specified in the request and that have the same parent as the new resource being created.
- If such a resource exists, then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "ORIGINATOR_HAS_ALREADY_REGISTERED" error.

7.4.5.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.5.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.5.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the with the following addition:

Primitive Specific operation on Recv-6.6 "Announce/De-announce the resource":

1) If the request is for an <AE> resource and if it is received at a MN-CSE or ASN-CSE with AE-ID-Stem starting with "S" the receiver shall execute the following steps in order:

a) Compose the Update <AEAnnc> Request primitive with the *link* attribute set to "INACTIVE".

b) Send Update request to the announced to IN-CSE.

c) Wait for Response primitive.

7.4.5.2.5 Notify

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.6 Resource Type <container>

7.4.6.1 Introduction

This resource represents a container for data instances. It is used to share information among other entities and potentially to track the data. A <container> resource has no associated content, only attributes and child resources.

The detailed description can be found in clause 9.6.6 in oneM2M TS-0001 [6].

Table 7.4.6.1-1: Data type definition of <container> resource

Data Type ID	File Name	Note
container	CDT-container-v3_11_0.xsd	

Table 7.4.6.1-2: Universal/Common Attributes of <container> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
stateTag	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.6.1-3: Resource Specific Attributes of <container> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
maxNrOfInstances	O	O	xs:nonNegativeInteger	No default
maxByteSize	O	O	xs:nonNegativeInteger	No default
maxInstanceAge	O	O	xs:nonNegativeInteger	No default
currentNrOfInstances	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the maxNrOfInstances)
currentByteSize	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the maxByteSize)
locationID	NP	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
disableRetrieval	O	O	xs:boolean	Default value is false, when the parameter is not specified

Table 7.4.6.1-4: Child resources of <container> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<contentInstance>	[variable]	0..n	Clause 7.4.7
<subscription>	[variable]	0..n	Clause 7.4.8
<container>	[variable]	0..n	Clause 7.4.6
<latest>	la	1	Clause 7.4.27
<oldest>	ol	1	Clause 7.4.28
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<flexContainer>	[variable]	0..n	Clause 7.4.37
<timeSeries>	[variable]	0..n	Clause 7.4.38
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.6.2 <container> resource specific procedures for CRUD operations

7.4.6.2.0 Introduction

This clause describes container resource specific behaviour for CRUD operations.

7.4.6.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The primitive-specific operation Recv-6.4 is performed with following exceptions for optional attributes while executing procedures defined in clause 7.3.3.3.

The Hosting CSE may assign default values based on local policy for optional attributes *maxNrOfInstances*, *maxByteSize* and *maxInstanceAge*.

If the *maxNrOfInstances*, *maxByteSize* or *maxInstanceAge* attributes are present in the resource representation, but their value indicates an invalid value, then the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

There are two cases where the Hosting CSE may configure or override a *maxNrOfInstances*, *maxByteSize* or *maxInstanceAge* value specified in the resource representation (if present).

1) If the Originator does not specify a value the Hosting CSE may configure a *maxNrOfInstances*, *maxByteSize* or *maxInstanceAge* into the resource according to local policy. If the Hosting CSE has configured a value it shall return this value back to the originator in the response if the ***Result Content*** parameter permits this.

2) If the Hosting CSE determines that the *maxNrOfInstances*, *maxByteSize* or *maxInstanceAge* requested by the Originator does not meet its requirements (e.g. based on a local policy) the Hosting CSE shall configure a *maxNrOfInstances*, *maxByteSize* or *maxInstanceAge* into the resource according to local policy. The Hosting CSE shall return the modified value back to the originator in the response if the ***Result Content*** parameter permits this.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.6.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.6.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.6.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.7 Resource Type <contentInstance>

7.4.7.1 Introduction

The <contentInstance> resource represents a data instance in the container.

The detailed description can be found in clause 9.6.7 in oneM2M TS-0001 [6].

Table 7.4.7.1-1: Data type definition of <contentInstance> resource

Data Type ID	File Name	Note
contentInstance	CDT-contentInstance-v3_11_0.xsd	

Table 7.4.7.1-2: Universal/Common Attributes of <contentInstance> resource

Attribute Name	Request Optionality
	Create
@resourceName	O
resourceType	NP
resourceID	NP
parentID	NP
expirationTime	O
creationTime	NP
lastModifiedTime	NP
stateTag	NP
labels	O
announceTo	O
announcedAttribute	O
creator	O

Table 7.4.7.1-3: Resource Specific Attributes of <contentInstance> resource

Attribute Name	Request Optionality	Data Type	Default Value and Constraints
	Create		
contentInfo	O	m2m:contentInfo	No default
contentSize	NP	xs:nonNegativeInteger	No default
contentRef	O	m2m:contentRef	No default
ontologyRef	O	xs:anyURI	No default
content	M	xs:anyType	No default (Transfer encoding may be applied, and indicated applied encoding as part of the *contentInfo* attribute)

Table 7.4.7.1-4: Child resources of <contentInstance> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<transaction>	[variable]	0..n	Clause 7.4.61

The *contentInfo* attribute shall provide meta information about the stored data in *content* and is optional. See the definition of m2m:contentInfo in Table 6.3.3-1: oneM2M Simple Data Types for details

7.4.7.2 <contentInstance> resource specific procedures for CRUD operations

7.4.7.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations.

1) The Hosting CSE shall check whether the size in bytes of the *content* attribute of the <contentInstance> resource is greater than *maxByteSize* of the targeted parent <container> resource.

   a) If true, the Hosting CSE shall return the response primitive with a ***Response Status Code*** indicating "NOT_ACCEPTABLE" error. Skip steps 2 and 3 below.

   b) If false, the Hosting CSE shall set the *contentSize* attribute of the <contentInstance> resource to the size in bytes of the *content* attribute.

2) The Hosting CSE shall check the *currentNrOfInstances* and *currentByteSize* of the targeted parent <container> resource.

   a) If *maxNrOfInstances* of the targeted parent <container> resource is specified then if the *currentNrOfInstances* when modified to reflect the addition of the new <contentInstance> exceeds *maxNrOfInstances*, the Hosting CSE shall remove the oldest <contentInstance> resource from the targeted <container> resource.

   b) If *maxByteSize* of the targeted parent <container> resource is specified then if the *currentByteSize* when modified to reflect the addition of the new <contentInstance> exceeds *maxByteSize* the Hosting CSE shall remove the oldest <contentInstance> resources from the targeted <container> resource until *maxByteSize* conditions are met.

   c) The Hosting CSE shall update the *currentNrOfInstances* of the targeted parent <container> resource with the count of <contentInstance> resources in the targeted parent <container> resource. The Hosting CSE shall update the *currentByteSize* of the targeted parent <container> resource with the sum of the *contentSize* attributes of the <contentInstance> resources in the targeted parent <container> resource.

   d) When removing the oldest <contentInstance> resources, the Hosting CSE shall not generate notifications even if there exists a <subscription> to the targeted <container> resource and this <subscription> is configured to generate a notification on "Delete_of_Direct_Child_Resource".

   e) If the *maxInstanceAge* attribute is present in the targeted parent <container> resource, then the Hosting CSE shall set the *expirationTime* attribute in <contentInstance> resource such that the time difference between *expirationTime* and the *creationTime* of the <contentInstance> resource shall not exceed the *maxInstanceAge* of the targeted parent <container> resource.

3) The Hosting CSE shall increment the *stateTag* attribute of the targeted parent <container> resource and copy the value into the *stateTag* attribute of the <contentInstance> resource.

4) If the hosting CSE has the capability to duplicate the actual data in semantic triples, it may decide whether to represent the *content* as semantic triples, depending on local policies/configurations. If the hosting CSE decides to do so, it shall execute the following actions: a) represent the actual data contained in the *content* attribute to semantic triples (e.g. RDF triples); b) create a <semanticDescriptor> child resource for the <contentInstance> resource with its descriptor attribute set to these semantic triples generated in a).

5) If the hosting CSE does not have the capability to duplicate the actual data in semantic triples complying with an ontology that it supports, this step will be skipped.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.7.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except following conditions:

- If the value of *disableRetrieval* attribute of the parent <container> resource is true, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

7.4.7.2.3 Update

Originator:

The <contentInstance> resource shall not be Updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.7.2.4 Delete

Originator:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) *currentNrOfInstances* and *currentByteSize* of direct parent <container> resource shall be updated.

Receiver:

No change from the generic procedures in clause 7.2.2.2 except following conditions:

- If the value of *disableRetrieval* attribute of the parent <container> resource is true, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

7.4.8 Resource Type <subscription>

7.4.8.1 Introduction

The <subscription> resource contains subscription information for its subscribed-to resource. The subscription resource is a child of the subscribed to resource.

The detailed description can be found in clause 9.6.8 in oneM2M TS-0001 [6].

Table 7.4.8.1-1: Data type definition of <subscription> resource

Data Type ID	File Name	Note
subscription	CDT-subscription-v3_11_0.xsd	

Table 7.4.8.1-2: Universal/Common Attributes of <subscription> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.8.1-3: Resource Specific Attributes of <subscription> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
eventNotificationCriteria	O	O	m2m:eventNotificationCriteria	Default behaviour is notification on Update_of_Resource
expirationCounter	O	O	xs:positiveInteger	No default
notificationURI	M	O	list of xs:anyURI	No default In this value, it may contain notification serialization type (i.e. xml, json, cbor) per target. This shall be applied only for the URL formatted target (c.f. resource ID). When the type is set, only one type indication shall be appended in the target as the key-value format with delimiter "?",. If the value already contains "?" character for application queries, the type information shall be appended with "&". The key shall be "ct" (content serialization type). Note that this serialization type is in lower cases. Examples: http://mydomain/notificationHandler?ct=json http://mydomain/notificationHandler?q=true&ct=json
groupID	O	O	xs:anyURI	No default
notificationForwardingURI	O	O	xs:anyURI	No default
batchNotify	O	O	m2m:batchNotify	No default
rateLimit	O	O	m2m:rateLimit	No default
preSubscriptionNotify	O	NP	xs:positiveInteger	No default
pendingNotification	O	O	m2m:pendingNotification	No default
notificationStoragePriority	O	O	xs:positiveInteger	No default
latestNotify	O	O	xs:boolean	No default
notificationContentType	O	O	m2m:notificationContentType	Default value is set to 'all attributes'
notificationEventCat	O	O	m2m:eventCat	No default
subscriberURI	O	NP	xs:anyURI	No default
associatedCrossResourceSub	O	O	m2m:listOfURIs	No default. This attribute shall only be modified by a <crossResourceSubscription> Hosting CSE. described in clause 7.4.58

Table 7.4.8.1-4: Reference of child resources

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<schedule>	notificationSchedule	0..1	Clause 7.4.9
<notificationTargetMgmtPolicyRef>	[variable]	0..n	Clause 7.4.30
<notificationTargetSelfReference>	ntsr	1	Clause 7.4.33
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.8.2 <subscription> resource specific procedures for CRUD operations

7.4.8.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following are additional Hosting CSE procedures to the generic resource handling procedures (Figure 7.2.2.2-1 in clause 7.2.2.2). The additional procedures shall be inserted from Recv-6.2 to Recv-6.5 as below.

Recv-6.2. The following step is in addition to the procedures defined in clause 7.3.3.1:

Check if the subscribed-to resource, addressed in ***To*** parameter in the Request, is subscribable. Subscribable resource types are defined in TS-0001 [6]; they have <subscription> resource types as their child resources. If it is not subscribable, the Hosting CSE shall return the Notify response primitive with a ***Response Status Code*** indicating "TARGET_NOT_SUBSCRIBABLE" error.

Recv-6.3 The following step is in addition to the procedures defined in clause 7.3.3.15:

Check if the Originator has privileges for retrieving the subscribed-to resource. If the Originator does not have the privilege, the Hosting CSE shall return the response primitive with ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

Recv-6.4 The following steps are in addition to the procedures defined in clause 7.3.3.3:

1) Check if the *notificationEventType* is set to "Blocking_Update".

   - If the subscribed-to resource already has a subscription with this *notificationEventType* the Hosting CSE shall return the response primitive with ***Response Status Code*** indicating "BLOCKING_SUBSCRIPTION_ALREADY_EXISTS" error if more than one notification of this type could be sent.

   - If there is more than one *notificationURI* specified, the Hosting CSE shall return the response primitive with ***Response Status Code*** indicating "BAD_REQUEST" error.

   - If any other attributes of the <subscription> resource are specified the Hosting CSE shall return the primitive with ***Response Status Code*** indicating "BAD_REQUEST" error.

   - If any condition tag of the *eventNotificationCriteria* attribute other than *attribute* condition tag is specified, the Hosting CSE shall return the response primitive with ***Response Status Code*** indicating "BAD_REQUEST" error.

2) If any of the *notificationURI* entries are not the Originator, the Hosting CSE may send a Subscription Verification request primitive to each of them as described in clause 7.5.1.2.3.

   a) If the Hosting CSE cannot send one or more Subscription Verification request primitives, the Hosting CSE shall return the Create <subscription> response primitive with a ***Response Status Code*** indicating "SUBSCRIPTION_VERIFICATION_INITIATION_FAILED" error.

   b) If the Hosting CSE sent all the Subscription Verification request primitives, the Hosting CSE shall check if each Notify response primitive contains a ***Response Status Code*** indicating "OK". If not, the Hosting

CSE shall return the Create <subscription> response primitive containing the ***Response Status Code*** indicating "SUBSCRIPTION_VERIFICATION_INITIATION_FAILED" error.

3) If the *associatedCrossResourceSub* is provided, check that the Hosting CSE ID value in the *associatedCrossResourceSub* is the same as the ***From*** parameter of the request. If not, return the response primitive with a ***Response Status Code*** indicating "BAD_REQUEST".

Recv-6.5: The following steps are in addition to the procedures defined in clause 7.3.3.5:

1) If the Originator provides a value of *childResourceType* which is not a valid child of the subscribed-to resource, the request shall be rejected with a "BAD_REQUEST" ***Response Status Code.***

2) If both the *notificationEventType* and *operationMonitor* are present in the Request, the request shall be rejected with a "BAD_REQUEST" ***Response Status Code.***

3) If the Originator does not provide *notificationContentType*, the Hosting CSE shall set it as "all attributes".

4) If the *notificationURI* is not the Originator, the Hosting CSE shall store the Originator ID as the <subscription> resource's *creator* attribute.

5) If the *batchNotify* attribute is present in the Request but *batchNotify*/*duration* is not provided by the Originator, the Hosting CSE shall set the value of *batchNotify*/*duration* to the default duration as given by the M2M Service Provider.

7.4.8.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.8.2.3 Update

Originator:

The following change from the generic procedures in clause 7.2.2.1.

Orig-1.0: The originator shall not specify *notificationEventType* set to "Blocking_Update".

Receiver:

The following are additional Hosting CSE procedures to the generic resource handling procedures in clause 7.2.2.2.

Recv-6.4: The following steps are in addition to the procedures defined in clause 7.3.3.4:

1) Check if the notificationEventType is set to "Blocking_Update". If so, the Hosting CSE shall return the response primitive with ***Response Status Code*** indicating "BAD_REQUEST" error.

Recv-6.5. The following steps are in addition to the procedures defined in clause 7.3.3.7:

1) If the Originator provides a value of *childResourceType* which is not a valid child of the subscribed-to resource,the request shall be rejected with a "BAD_REQUEST" ***Response Status Code***.

2) If the UPDATE operation would result in both *operationMonitor* and *notificationEventType* being present in the resource, the request shall be rejected with a "BAD_REQUEST" ***Response Status Code.***

3) Check if a new *associatedCrossResourceSub* is provided. If so, check that the Hosting CSE ID value in the *associatedCrossResourceSub* is the same as the ***From*** parameter of the request.

4) If a <crossResourceSubscription> Hosting CSE ID is removed from *associatedCrossResourceSub*, the Hosting CSE shall send a Notify request for Subscription Deletion using the procedures in clause 7.5.1.2.4 to the <crossResourceSubscription> Hosting CSE.

7.4.8.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following are changes to the receiver procedures described in clause 7.2.2.2.

Recv-6.5. The Hosting CSE shall send a Notify request for Subscription Deletion using the procedures in clause 7.5.1.2.4 to all <crossResourceSubscription> hosting CSEs indicated in *associatedCrossResourceSub.*

7.4.9 Resource Type <schedule>

7.4.9.1 Introduction

The <schedule> resource shall represent scheduling information in the context of its parent resource. If a <schedule> resource is not present as a child resource then there are no time-constraints on the context of its parent resource. An Originator shall have the same access control privileges to the <schedule> resource as it has to its parent resource.

The detailed <schedule> resource description can be found in clause 9.6.9 of the TS-0001 [6].

Table 7.4.9.1-1: Data type definition of <schedule> resource

Data Type ID	File Name	Note
schedule	CDT-schedule-v3_11_0.xsd	

Table 7.4.9.1-2: Universal/Common Attributes of <schedule> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
accessControlPolicyIDs	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.9.1-3: Resource Specific Attributes of <schedule> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
scheduleElement	M	O	m2m:scheduleEntries	No Default and shall not be blank.
networkCoordinated	O	O	xs:boolean	This attribute is only applicable when <schedule> is a child resource of <node>. Default value is false.

The *scheduleElement* attribute represents the list of scheduled execution times.

Each entry of the *scheduleElement* attribute shall consist of a line with 7 field values (see Table 7.4.9.1-4).

The time to be matched with the schedule pattern shall be interpreted in UTC timezone.

Table 7.4.9.1-4: Definition of m2m:scheduleEntry string format

Field Name	Range of values	Note
Second	0 to 59	
Minute	0 to 59	
Hour	0 to 23	
Day of the month	1 to 31	
Month of the year	1 to 12	
Day of the week	0 to 6	0 means Sunday
Year	2000 to 9999	

Each field value can be either an asterisk ('*': matching all valid values), an element, or a list of elements separated by commas(',').

An element shall be either a number, a range (two numbers separated by a hyphen '-') or a range followed by a step value. A step value (a slash '/' followed by an interval number) specifies that values are repeated over and over with the interval between them. For example, note "0-23/2″ in the Hour field is equivalent to "0,2,4,6,8,10,12,14,16,18,20,22″. A step value can also be used after an asterisk (e.g. "*/2").

EXAMPLE 1:

EXAMPLE: * 0-5 2,6,10 * * * *

If the parent resource is a <node>, the Hosting CSE will forward requests to an AE or CSE hosted on the corresponding node during the time windows 2:00-2:05, 6:00-6:05, and 10:00-10:05 every day.

End of EXAMPLE 1:

EXAMPLE 2:

EXAMPLE: * * 8-20 * * * *

If the parent resource is a <subscription>, the Hosting CSE will not send notifications for the subscribed-to event between the hours of 20:00 and 8:00 every day.

End of EXAMPLE 2:

EXAMPLE 3:

EXAMPLE: * * 0-23/2 * * * *

If the parent resource is a <node>, the Hosting CSE will forward requests to an AE or CSE hosted on the corresponding node for an hour every other hour of every day.

End of EXAMPLE 3:

EXAMPLE 4:

EXAMPLE: * * * * * */2 *

If the parent resource is a <node>, the Hosting CSE will forward requests to an AE or CSE hosted on the corresponding node on Sundays, Tuesdays, Thursdays and Saturdays (*/2 in the day of the week field is equivalent to 0,2,4,6).

End of EXAMPLE 4:

Table 7.4.9.1-5: Child resources of <schedule > resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.9.2 <schedule> resource specific procedures for CRUD operations

7.4.9.2.0 Introduction

This clause describes <schedule> resource specific behaviour for CRUD operations.

7.4.9.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following are changes to the receiver procedures described in clause 7.2.2.2:

1) Recv-6.5: The following steps are in addition to the generic Create procedures defined in clause 7.3.3.5:

   a) The request shall be rejected with the "CONTENTS_UNACCEPTABLE" ***Response Status Code*** if the target resource is not a <node> resource and if the *networkCoordinated* attribute is present in the request.

   b) The request shall be rejected with the "NOT_IMPLEMENTED" ***Response Status Code*** if *networkCoordinated* is true and the Hosting CSE does not support coordination with an Underlying Network.

7.4.9.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.9.2.3 Update

Originator:

The following are changes to the receiver procedures described in clause 7.2.2.1:

1) Recv-6.5: The following steps are in addition to the generic Update procedures defined in clause 7.3.3.7.

   a) The request shall be rejected with the "CONTENTS_UNACCEPTABLE" ***Response Status Code*** if the target resource is not a <node> resource and if the *networkCoordinated* attribute is present in the request.

   b) The request shall be rejected with the "NOT_IMPLEMENTED" ***Response Status Code*** if *networkCoordinated* is true and the Hosting CSE does not support coordination with an Underlying Network.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.9.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.10 Resource Type <locationPolicy>

7.4.10.1 Introduction

The <locationPolicy> resource represents the method for obtaining and managing geographical location information of an M2M Node. The detailed description can be found in the clause 9.6.10 in oneM2M TS-0001 [6].

Table 7.4.10.1-1: Data type definition of <locationPolicy> resource

Data Type ID	File Name	Note
locationPolicy	CDT-locationPolicy-v3_11_0.xsd	

Table 7.4.10.1-2: Universal/Common Attributes of <locationPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.10.1-3: Resource Specific Attributes of <locationPolicy> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
locationSource	M	NP	m2m:locationSource	No default
locationInformationType	O	O	m2m:locationInformationType	No default
locationUpdatePeriod	O	O	List of xs:duration	No default
locationTargetID	O	NP	m2m:locationTargetID	No default
locationServer	O	NP	xs:anyURI	No default
locationContainerID	NP	NP	xs:anyURI	No default
locationContainerName	O	NP	xs:string	No default
locationStatus	NP	NP	xs:string	No default
geographicalTargetArea	O	O	xs:anyType	No default
geofenceEventCriteria	O	O	m2m:geofenceEventCriteria	No default
authID	O	NP	m2m:externalID	No default
retrieveLastKnownLocation	O	O	xs:boolean	No default
locationUpdateEventCriteria	O	O	m2m:locationUpdateEventCriteria	No default

Table 7.4.10.1-4: Child resources of <locationPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<*subscription*>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.10.2 <locationPolicy> resource specific procedures for CRUD Operations

7.4.10.2.0 Introduction

This clause describes <locationPolicy> resource specific primitive behaviour for CRUD operations.

7.4.10.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following <locationPolicy> resource type specific procedures shall be performed after Recv-6.5 and before Recv-6.6 generic procedures.

1) The Hosting CSE shall create a <container> resource in which the actual location information will be stored. Then the Hosting CSE shall create a <locationPolicy> resource and shall fill in cross-references for both resources. Both of these resources shall be hosted locally on the Hosting CSE. The *locationContainerID* attribute of the <locationPolicy> resource shall contain the *resourceID* of the created <container> resource and the *locationID* attribute of the <container> resource shall contain the *resourceID* of the <locationPolicy> resource. The name of the created <container> resource shall be determined by the *locationContainerName* attribute if it is applicable.

2) Check the locationSource and locationUpdatePeriod attributes:

   a) If the *locationSource* attribute is set by 'Network Based' and *locationUpdatePeriod* attribute is set by any duration value (higher than 0 second), or if the *locationSource* attribute is set by 'Network Based' and *locationUpdatePeriod* attribute is zero or not defined and *locationUpdateEventCriteria* is LocationChange, then continue with the step 3.

   b) If the *locationSource* attribute is set by 'Device Based' and *locationUpdatePeriod* attribute is set by any duration value (higher than 0 second), then continue with the step 4.

   c) If the *locationSource* attribute is set by 'Sharing Based' and *locationUpdatePeriod* attribute is set by any duration value (higher than 0 second), then continue with the step 5.

   d) If the *locationUpdatePeriod* has more than one values, the first value in the list shall be used as the current location update period in step 3,4 and 5. In this case, based on the local context information of the Hosting CSE such as velocity, available memory, the Hosting CSE may choose one of the value out of the list to be the active location update period. If the device is moving at the high speed, it is expected that the location of the device would be update more frequently. The Hosting CSE would acquire the current velocity of the device and compare the value with some predefined value, depend on the result of the comparison, the Hosting CSE would choose a smaller value from the list if the current velocity is higher than the predefined value. Otherwise, the Hosting CSE would choose a larger value.

3) The Hosting CSE shall retrieve the *locationTargetID* and *locationServer* attributes from the stored <location Policy> resource.

   If the *locationServer* is absent in the Originator's request, the Hosting CSE shall either derive the *locationServer* value from the *locationTargetID* or be pre-provisioned with the identity of a Location Server.

   In case either the *locationTargetID* or *locationServer* attribute cannot be obtained, the Hosting CSE shall reject the request with the ***Response Status Code*** indicating "BAD_REQUEST" error. Then, the Hosting CSE shall transform the location-acquisition request into Location Server request, using the attributes stored in

<locationPolicy> resource. The Hosting CSE shall also provide default values for other required parameters (e.g. quality of position) in the Location Server request according to local policies.

If the request which requests the location information of the target device towards the Location Server crosses over the Mcn reference point, the Hosting CSE shall add the *authID* in to requester parameter of the Location Server request [28].

The Hosting CSE shall send this Location Server request to the location server which could use one of the two API interfaces: one is OMA Mobile Location Protocol [i.4] and OMA RESTful NetAPI for Terminal Location [28]. The other one is 3GPP Monitoring Event API for terminal location [51]. If *retrieveLastKnownLocation* is true, the Hosting CSE shall use the 3GPP Monitoring Event API only. If the requester parameter is present in the request, the Location Server shall verify whether the requester is authorized to request the location information. If the requester is not authorized, PolicyException (POL0002) will be returned. If the requester is permitted, then the location server performs positioning procedure based upon the Location Server request. Then continue with step 6.

Based on the period information, *locationUpdatePeriod* attribute, this step can be periodically repeated or the location server can only notify the Hosting CSE of location information that performs periodically.

NOTE 1: The location server performs the privacy control and only responds successfully if the positioning procedure is permitted.

NOTE 2: Detailed information on how the Location Server request message is converted into OMA RESTful NetAPI for Terminal Location message is described in clause G.2.

NOTE 3: Detailed information on how the Location Server request message is converted into 3GPP Monitoring Event API for terminal location message is described in clause G.3.

4) The Hosting CSE shall perform positioning procedure using location determination modules and technologies (e.g. GPS). Then continue with step 6.

Based on the period information, *locationUpdatePeriod* attribute, this step can be periodically repeated.

NOTE 4: The Hosting CSE can utilize the internal interface (e.g. System Call) to communicate with the modules and technologies. The detailed procedure is out of scope.

5) The Hosting CSE shall collect information of topology of M2M Area Network using <node> resource and find the closest Node from the Originator that has registered with the Hosting CSE and has location information. The closest Node is determined by the minimum hop based on the collected topology information.

   a) If the Hosting CSE can find the closest Node from the Originator, the location information of the closest Node shall be stored as the location information of the Originator into a <contentInstance> resource under the created <container> resource.

   b) If the Hosting CSE cannot find the closest Node from the Originator, the location information of the Hosting CSE shall be stored as the location information of the Originator into a <contentInstance> resource under the created <container> resource.

6) The Hosting CSE shall receive the corresponding response and transform it into a Response primitive.

   a) If the positioning procedure failed and *retrieveLastKnownLocation* is false, the Hosting CSE shall store a statusCode based on the error code in the *locationStatus* attribute in the created <locationPolicy> resource. If the positioning procedure failed and *retrieveLastKnownLocation* is true, the Hosting CSE shall repeat step 3), once only, but requesting the last known location from the Location Server using 3GPP Monitoring Event API [51].

   b) If the positioning procedure is successfully complete which means that the Hosting CSE acquires the location information, The Hosting CSE shall store the acquired location information into a <contentInstance> resource under the created <container> resource.

NOTE 5: The format of location information in the *content* of <contentInstance> is decided by the Location Server.

7.4.10.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.10.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.10.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The procedure of the Receiver written in the clause 7.2.2.2 shall be the same as initial steps. A following step is the <locationPolicy> resource type specific procedure for DELETE operation.

1) Once the <locationPolicy> resource is deleted, the Receiver shall delete the associated resources (e.g. <container>, <contentInstance> resources). If the ***locationSource*** attribute and the ***locationUpdatePeriod*** attribute of the <locationPolicy> resource have been set with appropriate values, the Receiver shall tear down the session. The specific mechanism used to tear down the session depends on the support of the Underlying Network and other factors.

7.4.11 Resource Type <delivery>

7.4.11.1 Introduction

In order to be able to initiate and manage the execution of data delivery in a resource-based manner, the resource type delivery is defined. This resource type shall be used for forwarding requests from one CSE to another CSE when the ***Delivery Aggregation*** parameter in the request is set to true. The detailed description can be found in clause 9.6.11 in oneM2M TS-0001 [6].

Table 7.4.11.1-1: Data type definition of <delivery> resource

Data Type ID	File Name	Note
delivery	CDT-delivery-v3_11_0.xsd	

Table 7.4.11.1-2: Universal/Common Attributes of <delivery> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
expirationTime	O	O
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
accessControlPolicyIDs	O	O
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.11.1-3: Resource Specific Attributes of <delivery> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
source	M	NP	m2m:ID	No default
target	M	NP	m2m:ID	No default
lifespan	M	O	m2m:timestamp	No default
eventCat	M	O	m2m:eventCat	No default
deliveryMetaData	M	O	m2m:deliveryMetaData	No default
aggregatedRequest	O	O	m2m:aggregatedRequest	No default

Table 7.4.11.1-4: Child resources of <delivery> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	variable	0..n	Clause 7.4.8

7.4.11.2 <delivery> resource specific procedures for CRUD operations

7.4.11.2.0 Introduction

This clause describes <delivery> resource specific behaviour for CRUD operations.

7.4.11.2.1 Create

Originator:

An AE shall not originate a Create <delivery> resource request.

Primitive specific operation on Orig-1.0 "Compose Request primitive":

1) The Originator shall use a blocking request (i.e. ***Response Type***=blockingRequest).

2) The Originator shall provide the content of the <delivery> resource.

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received over Mca reference point, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with a ***Response Status Code*** indicating 'OPERATION_NOT_ALLOWED' error.

   b) "Send the Response primitive".

2) Otherwise:

   a) No change from the generic procedures in clause 7.2.2.2.

NOTE: Determination of the reference point is at the discretion of the Receiver CSE implementation.

7.4.11.2.2 Retrieve

Originator:

Primitive specific operation on Orig-1.0 "Compose Request primitive":

1) The Originator shall use a blocking request (i.e. ***Response Type***=blockingRequest).

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.11.2.3 Update

Originator:

An AE shall not originate a Create <delivery> resource request.

Primitive specific operation on Orig-1.0 "Compose Request primitive":

1) The Originator shall use a blocking request (i.e. ***Response Type***=blockingRequest).

2) The Originator shall provide the content of the <delivery> resource.

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received over Mca reference point, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with a ***Response Status Code*** indicating 'OPERATION_NOT_ALLOWED' error.

   b) "Send the Response primitive".

2) Otherwise:

   a) No change from the generic procedures in clause 7.2.2.2.

NOTE: Determination of the reference point is at the discretion of the Receiver CSE implementation.

7.4.11.2.4 Delete

Originator:

An AE shall not originate a Create <delivery> resource request.

Primitive specific operation on Org-1.0 "Compose Request primitive":

1) The Originator shall use a blocking request (i.e. ***Response Type***=blockingRequest).

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received over Mca reference point, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with a ***Response Status Code*** indicating 'OPERATION_NOT_ALLOWED' error.

   b) "Send the Response primitive".

2) Otherwise:

   a) No change from the generic procedures in clause 7.2.2.2.

NOTE: Determination of the reference point is at the discretion of the Receiver CSE implementation.

7.4.12 Resource Type <request>

7.4.12.1 Introduction

The <request> resource is used to represent information relating to a non-blocking request. If an AE or CSE issues a request (including a notification request) targeting any resource other than a <request>, using a non-blocking mode (e.g. if the ***Response Type*** parameter of the request is set to "nonBlockingRequestSynch" or "nonBlockingRequestAsynch") and if the Registrar CSE of the Originator supports the <request> resource type, as indicated by the *supportedResourceType* attribute of the <CSEBase> resource representing the Registrar CSE of the Originator, the Registrar CSE of the Originator shall create an instance of <request> to capture and expose the context of the associated non-blocking request. More details can be found in clause 7.3.2 of the present document and in clause 9.6.12 in oneM2M TS-0001 [6].

Table 7.4.12.1-1: Data type definition of <request> resource

Data Type ID	File Name	Note
request	CDT-request-v3_11_0.xsd	

Table 7.4.12.1-2: Universal/Common Attributes of <request> resource

Attribute Name	Request Optionality
	Update
@resourceName	NP
resourceType	NP
resourceID	NP
expirationTime	NP
parentID	NP
creationTime	NP
lastModifiedTime	NP
accessControlPolicyIDs	O
labels	NP
dynamicAuthorizationConsultationIDs	O

Table 7.4.12.1-3: Resource Specific Attributes of <request> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
operation	NP	NP	m2m:operation	No default
target	NP	NP	xs:anyURI	No default
originator	NP	NP	m2m:ID	No default
requestID	NP	NP	m2m:requestID	No default
metaInformation	NP	NP	m2m:metaInformation	No default
primitiveContent	NP	NP	m2m:primitiveContent	No default
requestStatus	NP	NP	m2m:requestStatus	No default
operationResult	NP	NP	m2m:operationResult	No default

Table 7.4.12.1-4: Reference of child resources

Child Resource Type Name	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8

7.4.12.2 <request> resource specific procedures for CRUD operations

7.4.12.2.0 Introduction

This clause describes request resource specific procedure on Resource Hosting CSE for CRUD operations.

7.4.12.2.1 Create

Originator:

The <request> resource shall not be created via API. It is only created implicitly by a Receiver CSE. See clause 7.3.2.2 Create <request> resource locally.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) "Create an unsuccessful Response primitive" with a ***Response Status Code*** indicating 'OPERATION_NOT_ALLOWED' error.

2) "Send the Response primitive".

7.4.12.2.2 Retrieve

Originator:

The procedure of the Originator is the same as the clause 7.2.2.1, except that the Originator shall not use the "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" communication methods.

Receiver:

The generic operation Recv-2.0 "Communication method?" is performed with the following additions:

- If the ***Response Type*** parameter is "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" return an unsuccessful Response primitive with a ***Response Status Code*** indicating a "BAD_REQUEST" error.

- If the ***Response Type*** parameter is "flexBlocking" treat the request as if the ***Response Type*** were "blockingRequest".

7.4.12.2.3 Update

Originator:

The procedure of the Originator is the same as the clause 7.2.2.1, except that the Originator shall not use the "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" communication methods.

Receiver:

The generic operation Recv-2.0 "Communication method?" is performed with the following additions:

- If the ***Response Type*** parameter is "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" return an unsuccessful Response primitive with a ***Response Status Code*** indicating a "BAD_REQUEST" error.

- If the ***Response Type*** parameter is "flexBlocking" treat the request as if the ***Response Type*** were "blockingRequest".

7.4.12.2.4 Delete

Originator:

The procedure of the Originator is the same as the clause 7.2.2.1, except that the Originator shall not use the "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" communication methods.

An Originator is not required to delete a <request> resource explicitly, as the CSE hosting the <request> resource will do it implicitly, but an Originator can use Delete to attempt to cancel a request.

Receiver:

The generic operation Recv-2.0 "Communication method?" is performed with the following additions:

- If the ***Response Type*** parameter is "nonBlockingRequestSynch" or "nonBlockingRequestAsynch" return an unsuccessful Response primitive with a ***Response Status Code*** indicating a "BAD_REQUEST" error.

- If the ***Response Type*** parameter is "flexBlocking" treat the request as if the ***Response Type*** were "blockingRequest".

The generic operation Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation" is performed with the following additional operations:

- If the <request> resource's *requestStatus* is PENDING, the associated operation shall be cancelled if possible. If it is not possible to cancel the operation return an unsuccessful Response primitive with a ***Response Status Code*** indicating an "UNABLE_TO_RECALL_REQUEST" error and do not delete the <request> resource.

- If the <request> resource's *requestStatus* is FORWARDED or PARTIALLY_COMPLETED, return an unsuccessful Response primitive with a ***Response Status Code*** indicating an "UNABLE_TO_RECALL_REQUEST" error and do not delete the <request> resource.

7.4.13 Resource Type <group>

7.4.13.1 Introduction

The <group> resource represents a group of resources of the same or mixed types. The <group> resource can be used to do bulk manipulations on the resources represented by the *memberIDs* attribute. The <group> resource contains an attribute that represents the members of the group and a virtual resource (the <fanOutPoint>) that allows operations to be applied to the resources represented by those members. The detailed description can be found in clause 9.6.13 in oneM2M TS-0001 [6].

Table 7.4.13.1-1: Data type definition of <group> resource

Data Type ID	File Name	Note
group	CDT-group-v3_11_0.xsd	

Table 7.4.13.1-2: Universal/Common Attributes of <group> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.13.1-3: Resource Specific Attributes of <group> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
memberType	O	NP	m2m:memberType	Default value is set to 'MIXED'
specializationType	O	NP	m2m:specializationType	No default
currentNrOfMembers	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the *maxNrOfMembers* attribute of the <group> resource)
maxNrOfMembers	M	O	xs:positiveInteger	No default
memberIDs	M	O	list of xs:anyURI	No default This list may contain no members
membersAccessControlPolicyIDs	O	O	m2m:listOfURIs	No default
memberTypeValidated	NP	NP	xs:boolean	No default (This is generated by the Hosting CSE)
consistencyStrategy	O	NP	m2m:consistencyStrategy	Default value is set to 'ABANDON_MEMBER'
groupName	O	O	xs:string	No default
semanticSupportIndicator	NP	NP	xs:boolean	No default (This is generated by the Hosting CSE and the value shall be true when this attribute is present)
notifyAggregation	O	O	m2m:batchNotify	No default

Table 7.4.13.1-4: Child resources of <group> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<fanOutPoint>	fopt	1	Clause 7.4.14
<semanticFanOutPoint>	sfop	0..1	Clause 7.4.35
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.13.2 <group> resource specific procedures for CRUD operations

7.4.13.2.0 Introduction

This clause describes <group> resource specific procedure on Resource Hosting CSE for CRUD operations.

7.4.13.2.1 Create

Originator:

If an Originator is creating a <group> containing members which are themselves of type <group>, the Originator shall add the suffix '/fopt' to a member ID if the Originator wants to fan-out group requests to each member of that sub-<group>, else the Originator shall not suffix the '/fopt' to that member ID.

For a group of resources of the same *resourceType* the originator shall set the *memberType* attribute to the type of resource desired. If the originator wants to create a group of the same specialized type of <flexContainer> resource then *memberType* shall be set to “flexContainer” and *specializationType* shall be set to the *containerDefinition* for the specialized type. If the originator wants to create a group of the same specialization of <mgmtObj>, *memberType* shall be set to "mgmtObj" and *specializationType* shall be set to the *mgmtDefinition* for the specialized type. If the originator wants to create a group with a variety of specialized resources the *specializationType* attribute shall be empty.

Receiver:

Primitive-specific operations after Recv-6.4 "Check validity of resource representation for the given resource type" and before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed". See clause 7.2.2.2.

1) Validate the provided attributes. The receiver shall also check that the number of URIs present in the *memberIDs* attribute of the group resource representation does not exceed the maximum as specified by the *maxNrOfMembers* attribute. If the maximum is exceeded, the request shall be rejected with a ***Response Status Code*** indicating the "MAX_NUMBER_OF_MEMBER_EXCEEDED" error. If there are duplicate members in the *memberIDs* attribute then the duplicate members are removed before creation of the <group> resource. If the *memberType* attribute of the <group> resource is not "MIXED", the Hosting CSE shall also verify that all the member IDs, including sub-groups, in the attribute *memberIDs* of the <group> resource representation provided in the request shall conform to the *memberType* of the group resource. To validate a resource type of a member, the Hosting CSE shall check the *resourceType* attribute of the resource which is indicated by the member ID. If the *specializationType* attribute is set in the request, the Hosting CSE shall verify that all the member IDs, including sub-groups, in the *memberIDs* attribute of the <group> resource representation provided in the request conform to that *specializationType*. To check the *resourceType* and *specializationType* attributes, the Hosting CSE may retrieve the member resources. When a member is a virtual resource other than a <fanOutPoint>, the Hosting CSE shall check the *resourceType* attribute of the parent resource. If the resource type of the parent allows this child virtual resource type, the Hosting CSE checks whether the virtual resource type matches with the *memberType* attribute of the group. If they match, then the Hosting CSE considers that the virtual member resource is validated. If the *resourceType* cannot be retrieved due to lack of privilege, the request shall be rejected with a ***Response Status Code*** indicating the "RECEIVER_HAS_NO_PRIVILEGE" error.

2) In the case that the <group> resource contains fanOutPoints of sub-group member resources (i.e. a *memberID* of a sub-group member is suffixed with /fopt), the receiver shall retrieve the *memberType* of each sub-group member resources to validate the *memberType*. If the *memberType* cannot be retrieved due to lack of privilege, the request shall be rejected with a ***Response Status Code*** indicating "RECEIVER_HAS_NO_PRIVILEGE". If the sub-group member resources are temporarily unreachable, the receiver shall set the *memberTypeValidated* attribute of the <group> resource to false and return the result to the originator in the response of the request. As soon as any unreachable sub-group resource becomes reachable, the receiver shall perform the *memberType* validation procedure. 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 SET_MIXED.

3) The *memberTypeValidated* attribute shall be set to true if all the members have been validated successfully. If a member validation for the *memberType* of the <group> resource is unsuccessful, then the Hosting CSE shall perform the following:

    a) If the *consistencyStrategy* of the <group> resource is ABANDON_GROUP then the request shall be rejected with a ***Response Status Code*** indicating the "GROUP_MEMBER_TYPE_INCONSISTENT" error.

    b) If the *consistencyStrategy* of the <group> resource is ABANDON_MEMBER then remove the inconsistent members and create the <group> resource and the *memberTypeValidated* attribute shall be set to true.

    c) If the *consistencyStrategy* of the <group> resource is SET_MIXED then set the *memberType* attribute of the <group> resource to "MIXED" and create the <group> resource and the *memberTypeValidated* attribute shall be set to true.

After Recv-6.7 "Create a success response", the receiver shall perform multicast group creation procedure if the Group Hosting CSE supports multicast. See clause 7.5.3.1.

7.4.13.2.2 Retrieve

No primitive specific operations.

7.4.13.2.3 Update

Originator:

If the Originator intends to update the *memberIDs* attribute, and there are some members which are themselves of type <group>, the Originator shall add the suffix the '/fopt' to a member ID if the Originator wants to fan-out group requests to each member of that sub-<group>, else the Originator shall not suffix the '/fopt' to that member ID.

Receiver:

Primitive specific operations after Recv-6.4 "Check validity of resource representation for the given resource type" and before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed". See clause 7.2.2.2.

1) If the memberType attribute of the <group> resource is not "MIXED", the Hosting CSE shall verify that all the member IDs, including sub-groups, in the attribute memberIDs of the <group> resource representation provided in the request shall conform to the memberType of the <group> resource. Virtual member resource validation shall be done as specified in the group creation procedure (clause 7.4.13.2.1). If the *specializationType* attribute was set in the <group> resource, the Hosting CSE shall verify that all of the member IDs, including sub-groups, in the *memberIDs* attribute of the <group> resource representation provided in the request conform to that *specializationType*.

2) In the case that the <group> resource contains fanOutPoints of sub-group member resources (i.e. a *memberID* of a sub-group member is suffixed with /fopt), the Receiver shall retrieve the *memberType* of each sub-group member resource to validate the *memberType*. If the *memberType* cannot be retrieved due to lack of privilege, the request shall be rejected with a ***Response Status Code*** indicating "RECEIVER_HAS_NO_PRIVILEGE" error. If the sub-group member resources are temporarily unreachable, the receiver shall set the *memberTypeValidated* attribute of the <group> resource to false and return the result to the originator in the response of the request. As soon as any unreachable sub-group resource becomes reachable, the receiver shall perform the *memberType* validation procedure. The Originator may get to know the validation result by subscribing to the created resource if the *memberTypeValidated* attribute is false. 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 SET_MIXED.

3) The *memberTypeValidated* attribute shall be set to true if all the members have been validated successfully. If a member validation for the *memberType* of the <group> resource is unsuccessful, then the Hosting CSE shall perform the following:

    a) If the *consistencyStrategy* of the <group> resource is ABANDON_GROUP then the request shall be rejected with a ***Response Status Code*** indicating "GROUP_MEMBER_TYPE_INCONSISTENT" error.

    b) If the *consistencyStrategy* of the <group> resource is ABANDON_MEMBER then remove the inconsistent members and update the <group> resource and the *memberTypeValidated* attribute shall be set to true.

    c) If the *consistencyStrategy* of the <group> resource is SET_MIXED then set the *memberType* attribute of the <group> resource to "MIXED" and update the <group> resource and the *memberTypeValidated* attribute shall be set to true.

4) Primitive-specific operation: The Hosting CSE shall check whether the number of members in the request's *memberIDs* attribute exceeds the limitation of *maxNrOfMembers*. The Hosting CSE shall also check whether the value provided in the *maxNrOfMembers* attribute is smaller than the *currentNrOfMembers* attribute value. If any of the condition is true, the Hosting CSE shall reject the request with ***Response Status Code*** indicating "MAX_NUMBER_OF_MEMBER_EXCEEDED" error.

After Recv-10.0 "Send Response Primitive", the receiver shall perform multicast group UPDATE procedure if the Group Hosting CSE supports multicast. See clause 7.5.3.2.

7.4.13.2.4 Delete

After Recv-10.0 "Send Response Primitive", the receiver shall perform multicast group DELETE procedure if the Group Hosting CSE supports multicast. See clause 7.5.3.3.

7.4.14 Resource Type <fanOutPoint>

7.4.14.1 Introduction

The <fanOutPoint> resource is a virtual resource because it does not have a representation. It is the child resource of a <group> resource. Whenever a request is sent to the <fanOutPoint> resource, the request is fanned out to each of the members of the <group> resource indicated by the *memberIDs* attribute of the <group> resource. The responses (to the request) from each member are then aggregated and returned to the Originator. The detailed description can be found in clause 9.6.14 in oneM2M TS-0001 [6].

There are no common attributes, resource specific attributes or xsd file to <fanOutPoint> resource because it is a virtual resource.

A <fanOutPoint> is addressed in the following way:

- Using a hierarchical URI formed by taking the structured or unstructured resource identifier of the parent <group> and appending the string /fopt to that URI.

This hierarchical URI can be extended by appending further path elements beyond the place where /fopt/ occurs.
A request sent to such a URI is not fanned out to the group members, but instead it is fanned out to the resources located by taking the hierarchical URI of each group member in turn and then appending the additional path elements to that URI.

For example, if /IN-CSE-0001/myGroup were a group with members:

- /IN-CSE-0001/m1 and
- /IN-CSE-0001/m2

then a request sent to /IN-CSE-0001/myGroup/fopt/x/y would be fanned out to:

- /IN-CSE-0001/m1/x/y and
- /IN-CSE-0001/m2/x/y

The additional path elements can reference virtual resources, for example if m1 and m2 were both <container> resources then a request sent to /IN-CSE-0001/myGroup/fopt/la would be fanned out to the most recent <contentInstance> child resource of both m1 and m2.

If the memberIDs m1 and m2 are themselves the fanOutPoints of <group> resources (i.e. suffixed with /fopt), a request sent to /IN-CSE-0001/myGroup/fopt will be fanned out to all the members of m1 and all members of m2.

7.4.14.2 <fanOutPoint> operations

7.4.14.2.1 Validate the type of resource to be created

If this is a CREATE request and the *memberType* attribute of the addressed parent group resource is not "MIXED", the group Hosting CSE may check whether the type of resource to be created is a valid and compatible child resource type of the group members. If they are not consistent, the request shall be rejected with a ***Response Status Code*** indicating "INVALID_CHILD_RESOURCE_TYPE" error.

7.4.14.2.2 Sub-group creation for members residing on the same CSE

The group hosting CSE shall obtain URIs of addressed resources from the attribute *memberIDs* of the parent <group> resource. The group hosting CSE may determine that multiple member resources belong to the same remote member hosting CSE, and may act as an Originator to request to create a sub-group containing the specific multiple member resources in that member hosting CSE. This sub-group is created in the member hosting CSE as described in clause 7.4.13.2.1. The ***To*** parameter of this group Create request shall be <memberHosting cseBase>/<groupHosting remoteCse>/ or <memberHosting cseBase>/. The group hosting CSE shall also provide ***From*** parameter (i.e. group hosting CSE) and sub-group resource representation that contains a *memberIDs* attribute with all the members residing on the addressed member Hosting CSE. The sub-group representation may include the attribute

accessControlPolicyIDs, so that both the group hosting CSE and all permissions of the original group apply to this sub-group. The ID of the sub-group may be proposed by the group hosting CSE and accepted by the member hosting CSE or it may be given by the member hosting CSE.
If there is already a sub-group resource defined in the remote member hosting CSE, then the group hosting CSE may utilize the existing sub-group resource.

7.4.14.2.3 Assign URI for aggregation of notification

If the request is to create a <subscription> resource, the group hosting CSE shall validate the request to check whether it contains a *notificationForwardingURI* attribute or not. If it does not, the group hosting CSE shall forward it to the group members. If it does, the group hosting CSE shall assign a new URI to the *notificationURI* attribute of the <subscription> in the requests before forwarding it to the group members. This new URI shall address the group hosting CSE so that it can receive and aggregate Notifications from those subscriptions.

7.4.14.2.4 Fan out Request to each member

If the parent group has no members, the group hosting CSE shall reject the request with the ***Response Status Code*** indicating "NO_MEMBERS".

If the request contains a ***Group Request Target Members*** parameter, and if any of the member IDs in this parameter is not present in the *memberIDs* list of the parent group or any of its sub-group's *memberIDs* lists then the request shall be rejected with BAD_REQUEST ***Response Status Code.*** Otherwise the group hosting CSE shall fan out the request to members contained in the ***Group Request Target Members*** parameter only.

The group Hosting CSE shall perform the following steps for each member:

a) The primitive parameters ***From*** and ***To*** shall be mapped to the primitive parameters of the corresponding Request to be sent out to each member of the group. The primitive parameter ***From*** shall be directly used. The primitive parameter ***To*** (i.e. <URI of group resource>/fopt) shall be replaced by resource identifiers present in the ***memberIDs*** attribute of the group resource. Any additional relative address that was appended to .../fopt in the original Request shall be appended to each ***To*** URI. The group hosting CSE shall execute "Compose Request primitives". In addition, the group hosting CSE shall generate a unique group request identifier, add it as a primitive parameter to the Request and locally store the group request identifier as per the local policy.

b) "Send the Request to the receiver CSE".

c) "Wait for Response primitives".

The procedures between group hosting CSE and member hosting CSEs shall comply with the corresponding creation procedures as described in clause 7. The detailed procedures are according to the type of Resource provided in the Request primitive. During <fanOutPoint> manipulation, the member hosting CSE receiving a Request sent from the group hosting CSE shall check if the Request contains a ***Group Request Identifier*** parameter. If the Request contains a ***Group Request Identifier*** parameter, the member hosting CSE shall compare the ***Group Request Identifier*** parameter to the ***Group Request Identifier*** stored locally. If a match is found, the member hosting CSE shall reject the request with the ***Response Status Code*** indicating "GROUP_REQUEST_IDENTIFIER_EXISTS" error in the Response primitive. Otherwise, the member hosting CSE shall continue with the operations according to the Request and store the ***Group Request Identifier*** parameter locally.

7.4.14.2.5 Aggregation of member responses

After receiving the member responses from the member hosting CSEs, the group hosting CSE shall respond to the Originator with an aggregated response. To indicate which response is generated by which member resource, the Hosting CSE shall set that member's resource ID into the ***From*** response parameter in each member response.

If ***Response Type, Result Expiration Time*** or ***Result Persistence*** were set in the request, these affect the behaviour of the group hosting CSE as follows:

- If ***Response Type*** is set to **blockingRequest**, the group hosting CSE shall respond only once with the aggregated response. It shall do this before the time indicated by the ***Result Expiration Time*** is reached. The group hosting CSE shall discard any member responses received after this time.

- If ***Response Type*** is set to **nonBlockingRequestSynch**, the group hosting CSE shall create a <request> resource locally and respond the Originator with the address of this <request> resource. Until the ***Result Expiration Time*** is reached**,** the group hosting CSE shall aggregate the member responses and include this aggregated response in the *operationResult* of the <request> resource.

- If ***Response Type*** is set to **nonBlockingRequestAsynch**, the group hosting CSE shall notify the Originator or the notification targets with aggregated responses before the ***Result Expiration Time*** expires. The group hosting CSE may notify the Originator more than once during the period until the ***Result Expiration Time*** expires. Each notification shall contain different member responses.

- If ***Response Type*** is set to **flexBlocking**, the group hosting CSE shall keep aggregating the member responses until the group hosting CSE determines that it is time to send a response – this depends on the properties of the group hosting CSE related to the <group> resource (the number of aggregated responses or the time period of the aggregation). By that time, if the aggregated response contains all the member responses, the group hosting CSE shall respond with the aggregated response. However if only some of the member responses have been received, the group hosting CSE shall create a <request> resource from the received request, and respond to the Originator with the reference to the created <request> resource as well as the currently aggregated responses. Until the time specified in ***Result Expiration Time*** is reached, the group hosting CSE shall keep aggregating the remaining member responses and updating the aggregated response in the *operationResult* of the <request> resource. If **notificationTarget** is provided in the request, the group hosting CSE shall notify the Originator with the aggregated response. Each notification shall contain different member responses.

If the group hosting CSE supports <request> resource, in the **nonBlockingRequestAsynch**, **nonBlockingRequestSynch** and **flexBlocking** case, it shall set the *requestStatus* of the <request> resource to PARTIALLY_COMPLETED if some of the member responses are received. If the group hosting CSE has aggregated all the member responses, it shall set the *operationResult* to COMPLETED.

In any of the cases above, member responses received after the ***Result Expiration Time*** shall be discarded. After the time specified in ***Result Persistence***, the aggregated response shall not be retrievable.

If the group Hosting CSE gets no response before the ***Result Expiration Timestamp*** expiry, then the Hosting CSE shall return error with the ***Response Status Code*** parameter set as "GROUP_MEMBERS_NOT_RESPONDED". Otherwise, the group Hosting CSE shall return the successful ***Response Status Code*** parameter value "OK" regardless of the requested operation. Note that the "OK" successful ***Response Status Code*** parameter is set regardless of the ***Response Status Code*** parameter value in each response primitive from the group member(s).

When aggregating notifications the group hosting CSE, upon receiving the first notification, shall use the group's *notifyAggregation* attribute and wait until *notifyAggregation*/*number* notifications have been received or until the *notifyAggregation*/*duration* has elapsed, whichever comes first, and send a Notify primitive containing the aggregatedNotification data object defined in Table 7.5.1.1-2. If the *notifyAggregation* attribute is not specified in the <group> resource then the group hosting CSE shall use the *currentNrOfMembers* attribute of the <group> and a duration specified by the M2M Service Provider instead of the number and duration from the *notifyAggregation* attribute.

If any of the parameters mentioned above are missing from the request, the group hosting CSE shall determine the time to respond using its local Policy.

7.4.14.2.6 Multicast fan out procedure

The Group Hosting CSE shall perform the following steps. The procedure refers to clause 10.2.7.13.2 of oneM2M TS-0001 [6] for details.

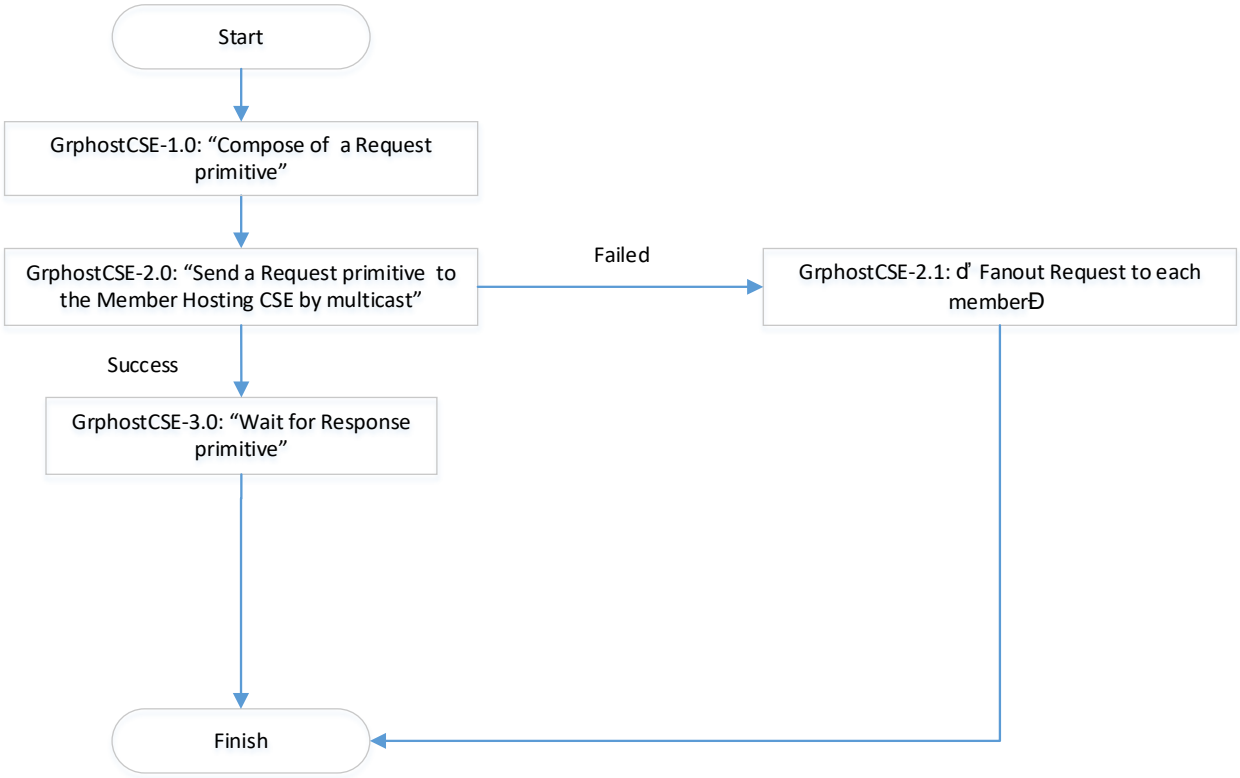

Figure 7.4.14.2.6-1: Generic procedure of Group Hosting CSE

GrphostCSE-1.0 "Compose a Request primitive": The Group Hosting CSE shall prepare the primitive parameters according to the Multicast Group Information: ***To*** shall be *multicastGroupFanoutTarget*, ***Request Expiration Timestamp*** shall be ***Request Expiration Timestamp*** in the request from the Originator. If the ***Request Expiration Timestamp*** was not set in the request, ***Request Expiration Timestamp*** shall be set as the responseTimeWindow according to the local policy. The ***Response Type*** shall not be included in the request primitive.

GrphostCSE-2.0 "Send a Request primitive to the Member Hosting CSE by multicast": The Group Hosting CSE shall check the multicastType in the Multicast Group Information: if the multicastType is MBMS, follow the 3GPP MBMS fanout procedure, refer to clause 7.4.14.2.7; if the multicastType is IP, send the Request to the multicastAddress in the Multicast Group Information.

GrphostCSE-2.1 "Fanout Request to each member": The Group Hosting CSE shall perform Fanout Request to each member, refer to clause 7.4.14.2.4.

GrphostCSE-3.0 "Wait for Response primitive": If the Group Hosting CSE receives responses from the multicast group member hosting CSEs before the ***Request Expiration Timestamp*** expiry, the group hosting CSE shall get their ***Request Identifier*** and ***From*** parameters and match them against the multicast fan out request: if the ***Request Identifier*** **i**s the same as the parameter in the request, and the ***From*** is the same as the member Hosting CSE ID of the multicast group, the group hosting CSE shall accept the response of multicast fan out.

The member Hosting CSE shall perform the following primitive specific operations:

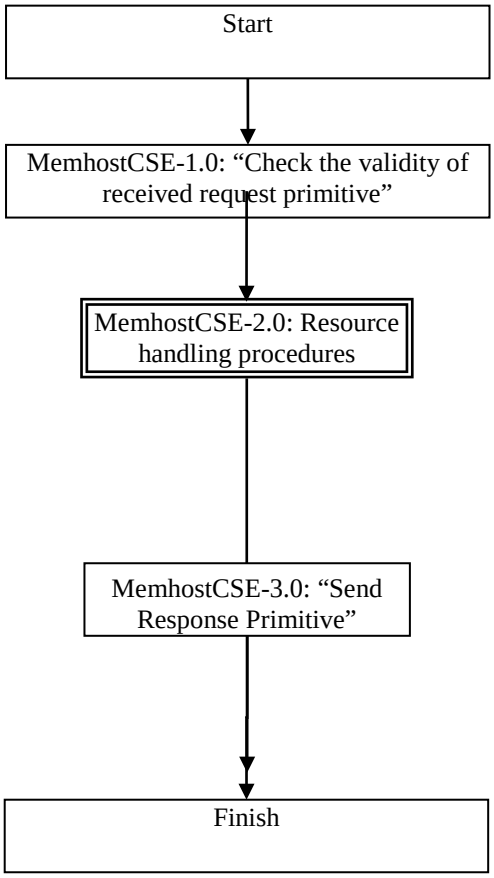

Figure 7.4.14.2.6-2: Generic procedure of Member Hosting CSE

MemhostCSE-1.0 "Check the validity of received request primitive": The Member Hosting CSE shall check the ***To*** primitive parameter in the request from the multicastAddress which it joined, and compare the ***To*** with the *multicastGroupFanoutTarget attribute* of all the <localMulticastGroup> resources to get the *memberList*. If there is no match result, the Member Hosting CSE shall return an error response with ***Response Status Code*** indicating "NOT_FOUND"; if there is a match result, the Member Hosting CSE shall replace the primitive parameter ***To*** by resource identifiers present in the *memberList* attribute of the local multicast group resource.

MemhostCSE-2.0: Resource handling procedures: Refer to Figure 7.2.2.2-2 for details.

MemhostCSE-3.0 "Send Response Primitive": The Member Hosting shall aggregate the operation results if there are multiple member IDs hosted on the same Member Hosting CSE, set the ***To*** primitive parameter as the value of *responseTarget* of the resource <localMulticastGroup> if the *responseTarget* exists. Set the ***From*** primitive parameter as the value of CSE-ID of the member Hosting CSE, and wait a randomized time that is less than the value of the *responseTimeWindow* of the resource <localMulticastGroup> if *responseTimeWindow* exists.

7.4.14.2.7 3GPP™ MBMS fan out procedure

The procedure is specified in the clause 7.7.3.2 in oneM2M TS-0026 [43].

1) The Group Hosting CSE shall check the TMGIExpiration in the Multicast Group Information: if the TMGIExpiration expires, the Group Hosting CSE shall execute the following steps in order:

   a) Send 3GPP Allocate TMGI Request [51] to the groupServiceServerAddress over Mcn reference point.

   b) Receive the corresponding response from 3GPP network: if the procedure completes successfully the Group Hosting CSE shall get the TMGI and TMGIExpiration from the response, then go to 2). If the procedure fails, the Group Hosting CSE shall perform a Fanout Request to each member in the Multicast Group Information, refer to clause 7.4.14.2.4.

2) If the TMGIExpiration does not expire, the Group Hosting CSE shall execute the following steps in order:

   a) Check the existing <*schedule*> child resources for all the Member Hosting CSE <*Node*> resources. If there is no time intersection of the existing <*schedule*>s, then return an error response with ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_REACHABLE" to the originator after which the procedure is terminated. If there is the time intersection, the Group Hosting CSE shall check if the ***Operation Execution Time*** or ***Request Expiration Timestamp*** is in the scope of the intersection when ***Operation Execution Time*** or ***Request Expiration Timestamp*** is included in the request, If ***Operation Execution Time*** is not in the scope of the intersection, the Group Hosting CSE shall return an error with ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_RQET_TIMEOUT" after which the procedure is terminated. If ***Request Expiration Timestamp*** is not in the scope of the intersection, the Group Hosting CSE shall return an error with ***Response Status Code*** indicating "EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_OET_TIMEOUT" after which the procedure is terminated. Otherwise, go to b).

   b) Send 3GPP Group Message Delivery via MBMS [51] to the groupServiceServerAddress in the Multicast Group Information over the Mcn reference point.

   c) Receive the corresponding response from 3GPP network: if the procedure fails or the parameter in the result indicates the delivery to some members failed, the Group Hosting CSE shall perform Fan out Request to each failed member, refer to clause 7.4.14.2.4.

7.4.14.3 <fanOutPoint> resource specific procedures for CRUD operations

7.4.14.3.1 Introduction

This clause describes <fanOutPoint> resource specific behaviour for CRUD operations.

7.4.14.3.2 Create

A Create operation sent to a <fanOutPoint> is fanned out to the members (if any) of the parent <group>. It is equivalent to sending a Create to each member and therefore results in new resources being created as children of these existing members.

If the Create is sent to a hierarchical URI containing a fanOutPoint and an additional path relative to that fanOutPoint then the new resources are not created as immediate children of the members, rather they are created as children of descendants of those members (as determined by the relative path).

Originator:

Primitive specific operation after Orig-1.0 "Compose Request primitive" and before Orig-2.0 "Send the Request to the receiver CSE": In the case the Originator wants to subscribe to all the member resources of the group and the originator wants the group hosting CSE to aggregate all the notifications come from its member hosting CSEs, the Originator shall include ***notificationForwardingURI*** attribute in the <subscription> resource.

If one or more of the individual responses has a ***Response Status Code*** other than CREATED then the Originator may issue a new CREATE request to the group Hosting CSE with the ***Group Request Target Members*** parameter containing the list of members for which the first CREATE failed.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 "Check authorization of the Originator".

Primitive specific operation additional to Recv-6.3 "Check authorization of the Originator": The Group Hosting CSE shall check the authorization of the Originator based on the *membersAccessControlPolicyIDs* of the parent <group> resource. In the case the *membersAccessControlPolicyIDs* is not provided, the *accessControlPolicyIDs* of the parent <group> resource shall be used.

Primitive specific operation to replace Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and Recv-6.6 "Announce/De-announce the resource" in the generic procedure:

1) Validate the type of resource to be created, refer to clause 7.4.14.2.1.

2) If the members are not in the Multicast Group Information locally, then perform sub-group creation for members residing on the same CSE, refer to clause 7.4.14.2.2.

3) Assign URI for aggregation of notification, refer to clause 7.4.14.2.3.

4) If there is Multicast Group Information held locally, the receiver shall perform the Multicast fan out procedure, refer to clause 7.4.14.2.6. For the members which are not in the multicast group, the receiver shall perform the fan out Request to each member, refer to clause 7.4.14.2.4.

5) Aggregation of member responses, refer to clause 7.4.14.2.5.

7.4.14.3.3 Retrieve

Originator:

No primitive specific operations.

If one or more of the individual responses has a ***Response Status Code*** other than OK then the Originator may issue a new RETRIEVE request to the group Hosting CSE with the ***Group Request Target Members*** parameter containing the list of members for which the first RETRIEVE failed.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 "Check authorization of the Originator".

Primitive specific operation additional to Recv-6.3 "Check authorization of the Originator": The Group Hosting CSE shall check the authorization of the Originator based on the *membersAccessControlPolicyIDs* of the parent group resource. In the case the *membersAccessControlPolicyIDs* is not provided, the *accessControlPolicyIDs* of the parent <group> resource shall be used.

Primitive specific operation to replace Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and Recv-6.6 "Announce/De-announce the resource" in the generic procedure:

1) If the members are not in the Multicast Group Information held locally, then perform sub-group creation for members residing on the same CSE, refer to 7.4.14.2.2.

2) If there is Multicast Group Information held locally, the receiver shall perform the Multicast fan out procedure, refer to clause 7.4.14.2.6. For the members which are not in the multicast group, the receiver shall perform the fan out Request to each member, refer to clause 7.4.14.2.4.

3) Aggregation of member responses, refer to clause 7.4.14.2.5.

7.4.14.3.4 Update

Originator:

No primitive specific operations.

If one or more of the individual responses has a ***Response Status Code*** other than UPDATED then the Originator may issue a new UPDATE request to the group Hosting CSE with the ***Group Request Target Members*** parameter containing the list of members for which the first UPDATE failed.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 "Check authorization of the Originator".

Primitive specific operation additional to Recv-6.3 "Check authorization of the Originator": The Group Hosting CSE shall check the authorization of the Originator based on the *membersAccessControlPolicyIDs* of the parent group resource. In the case the *membersAccessControlPolicyIDs* is not provided, the *accessControlPolicyIDs* of the parent <group> resource shall be used.

Primitive specific operation to replace Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and Recv-6.6 "Announce/De-announce the resource" in the generic procedure:

1) If the members are not in the Multicast Group Information held locally, then perform sub-group creation for members residing on the same CSE, refer to clause 7.4.14.2.2.

2) If there is Multicast Group Information held locally, the receiver shall perform the Multicast fan out procedure, refer to clause 7.4.14.2.6. For the members which are not in the multicast group, the receiver shall perform the fan out Request to each member. See clause 7.4.14.2.4.

3) Aggregation of member responses, refer to clause 7.4.14.2.5.

7.4.14.3.5 Delete

The primitive deletes the member resources belonging to an existing <group> resource, along with their child resources.

Originator:

No primitive specific operations.

If one ore more of the individual response has a ***Response Status Code*** other than DELETED then the Originator may issue a new DELETE request to the group Hosting CSE with the ***Group Request Target Members*** parameter containing the list of members for which the first DELETE failed.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and Recv-6.3 "Check authorization of the Originator": The ***To*** parameter consists of the URI of the group resource plus a suffix consisting of /fopt or /fopt/ plus any additional appended relative address.

Primitive specific operation additional to Recv-6.3 "Check authorization of the Originator": The Group Hosting CSE shall check the authorization of the Originator based on the *membersAccessControlPolicyIDs* of the parent group resource. In the case the *membersAccessControlPolicyIDs* is not provided, the *accessControlPolicyIDs* of the parent <group> resource shall be used.

Primitive specific operation to replace Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and Recv-6.6 "Announce/De-announce the resource" in the generic procedure:

1) If the members are not in the Multicast Group Information held locally, then perform sub-group creation for members residing on the same CSE, refer to clause 7.4.14.2.2.

2) If there is Multicast Group Information held locally, the receiver shall perform the Multicast fan out procedure, refer to clause 7.4.14.2.6. For the members which are not in the multicast group, the receiver shall perform the fan out Request to each member. See clause 7.4.14.2.4.

3) Aggregation of member responses, refer to clause 7.4.14.2.5.

4) If one or more members respond with a DELETED Response Status Code, the <group> hosting CSE shall remove the references to them from the *membersIDs* attribute of the targeted <group>, unless the reference in question is a virtual resource reference.

7.4.15 Resource Type <mgmtObj>

7.4.15.1 Introduction

The <mgmtObj> resource contains management data which represents individual M2M management functions. It represents a general structure to map to technology specific data models. There are multiple specializations of <mgmtObj>; these are defined in the Annex D. Each of these specializations has its own schema file. There is no separate schema file just for <mgmtObj>, however the XML schema types for the specializations all conform to the pattern described in this clause.

Table 7.4.15.1-1: Universal/Common Attributes of <mgmtObj> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.15.1-2: Resource Specific Attributes of <mgmtObj> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	m2m:mgmtDefinition	No default
objectIDs	O	NP	list of xs:anyURI	No default
objectPaths	O	NP	list of xs:anyURI	No default
description	O	O	xs:string	No default
mgmtSchema	O	NP	xs:anyURI	No default. This attribute shall be present if *mgmtDefinition*=0 "Self-defined". It shall not be present otherwise.
[objectAttribute]	O	O	Name and data type are defined in Annex D or XSD file identified by the value of mgmtSchema attribute.	No default.
mgmtLink	O	O	m2m:mgmtLinkRef	No default. This attribute is only present if defined in Annex D.

Table 7.4.15.1-3: Child resources of <mgmtObj> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.15.2 <mgmtObj> resource specific procedures for CRUD operations

7.4.15.2.1 Introduction

This clause describes <mgmtObj> resource specific procedure on resource Hosting CSE for CRUD operations.

The procedures are defined for management when technology specific protocols are used. When service layer management is performed, generic procedures defined in clause 7.2.2 shall comply for resource creation, update, retrieval and deletion. Procedures additional to resource manipulations to perform the management are further defined in Annex D.

7.4.15.2.2 Create

Originator:

Primitive specific operation before Orig-1.0 "Compose Request primitive":

1) If the originator is the managed entity, it shall generate the <mgmtObj> resource representation based on the technology-specific data model object of the managed entity to be exposed. The *objectIDs* and *objectPaths* attributes may be set in the Request.

Receiver:

The following steps shall be performed after Recv-6.4 "Check validity of resource representation for the given resource type" and before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) "Identify the managed entity and the technology-specific protocol". See clause 7.3.4.1.

   The receiver shall generate the technology-specific data model object to be added to the managed entity based on the <mgmtObj> resource representation provided in the Request primitive. The receiver may determine the target location on the managed entity where the generated technology specific data model object shall be added based on the *objectIDs* and *objectPaths* provided in the request primitive and the technology-specific data model being used. The receiver may also choose to let the managed entity decide the target location where the generated technology specific data model object shall be added using technology specific mechanism.

2) "Establish a management session with the managed entity". See clause 7.3.4.3.

3) "Send the management request(s) to the managed entity corresponding to the received Request primitive". See clause 7.3.4.4.
   If the receiver receives an error response from the managed entity because the technology-specific data model object to be added already exists on the managed entity, the receiver shall check (e.g. by using the OMA-DM Get command) if the existing technology-specific data model object is the same as the one to be added, then it shall consider the requested primitive as successfully performed instead of sending an error response primitive; otherwise, it shall reject the request with the ***Response Status Code*** indicating an "ALREADY_EXISTS" error in the Response primitive. The receiver shall also record the location where the technology-specific data model object is added to the managed entity in the successful case. The *objectIDs* and *objectPaths* attributes may be set in the Request.

4) The receiver may repeat step 3 in order to add to the managed entity the technology-specific data model objects that are mapped from the mandatory sub-resources (including any descendants) that are required to be created automatically with the default attribute values.

7.4.15.2.3 Retrieve

Receiver:

Primitive specific operation after Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and before Recv-6.6 "Announce/De-announce the resource":

1) "Identify the managed entity and the technology-specific protocol". See clause 7.3.4.1.

2) "Locate the technology-specific data model objects to be managed on the managed entity". See clause 7.3.4.2.

3) "Establish a management session with the managed entity". See clause 7.3.4.3.

4) "Send the management request(s) to the managed entity corresponding to the received Request primitive". See clause 7.3.4.4. The receiver may also update the <mgmtObj> resource representation with the retrieved technology-specific data model object if required according to the local policy.

7.4.15.2.4 Update

The Update primitive is used for the update of the resource as well as the execution of a management procedure. The execution is performed using an Update primitive by addressing specific attribute to start the management procedure (see Annex D). If the Update primitive addresses both normal as well as executable attributes then it shall perform the execution after the update of normal attributes.

Receiver:

Primitive specific operation after Recv-6.4 "Check validity of resource representation" and before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) "Identify the managed entity and the technology-specific protocol". See clause 7.3.4.1.

2) "Locate the technology-specific data model objects to be managed on the managed entity". See clause 7.3.4.2.

3) "Establish a management session with the managed entity". See clause 7.3.4.3.

4) "Send the management request(s) to the managed entity corresponding to the received Request primitive". The receiver may also update the <mgmtObj> resource representation with the retrieved technology-specific data model object information if required according to the local policy. See clause 7.3.4.4.

7.4.15.2.5 Delete

Receiver:

Primitive specific operation after Recv-6.4 "Check validity of resource representation" and before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) "Identify the managed entity and the technology-specific protocol". See clause 7.3.4.1.

2) "Locate the technology-specific data model objects to be managed on the managed entity". See clause 7.3.4.2.

3) "Establish a management session with the managed entity". See clause 7.3.4.3.

4) "Send the management request(s) to the managed entity corresponding to the received Request primitive". See clause 7.3.4.4.

7.4.16 Resource Type <mgmtCmd>

7.4.16.1 Introduction

The <mgmtCmd> resource represents a method to execute management procedures or to model commands and remote procedure calls (RPC) required by existing management protocols and enables AEs to request management procedures to be executed on a remote entity. The detailed description can be found in clause 9.6.16 in oneM2M TS-0001 Architecture TS [6].

Table 7.4.16.1-1: Data type definition of <mgmtCmd> resource

Data Type ID	File Name	Note
mgmtCmd	CDT-mgmtCmd-v3_11_0.xsd	

Table 7.4.16.1-2: Universal/Common Attributes of <mgmtCmd> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.16.1-3: Resource Specific Attributes of <mgmtCmd> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
description	O	O	xs:string	size: 256 No default
cmdType	M	NP	m2m:cmdType	RESET, REBOOT, UPLOAD, DOWNLOAD, SOFTWAREINSTALL, SOFTWAREUPDATE, SOFTWAREUNINSTALL No default
execReqArgs	O	O	m2m:execReqArgsListType	A list of entries which are dependent on *cmdType*: If *cmdType*=RESET, *execReqArgsList*=resetArgsType. If *cmdType*=REBOOT, *execReqArgsList*=rebootArgsType. If *cmdType*=UPLOAD, *execReqArgsList*=uploadArgsType. If *cmdType*=DOWNLOAD, *execReqArgsList*=downloadArgsType. If *cmdType*=SOFTWAREINSTALL, *execReqArgsList*=softwareInstallArgsType. If *cmdType*=SOFTWAREUPDATE, *execReqArgsList*=softwareUpdateArgsType. If *cmdType*= SOFTWAREUNINSTALL, *execReqArgsList*=softwareUninstallArgsType. No default
execEnable	O	O	xs:boolean	No default
execTarget	M	O	m2m:nodeID	No default
execMode	O	O	m2m:execModeType	IMMEDIATEONCE, IMMEDIATEREPEAT, RANDOMONCE, RANDOMREPEAT Default=IMMEDIATEONCE
execFrequency	O	O	xs:duration	No default
execDelay	O	O	xs:duration	Default=0
execNumber	O	O	xs:nonNegativeInteger	Default=1

Table 7.4.16.1-4: Child resources of <mgmtCmd> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<execInstance>	[variable]	0..n	Clause 7.4.17
<transaction>	[variable]	0..n	Clause 7.4.61

The <mgmtCmd> shall be executed for the following modes:

- If *execMode* is IMMEDIATEONCE, <mgmtCmd> shall be executed immediately and only once. In this mode, *execFrequency*, *execDelay*, and *execNumber* shall not be used.
- If *execMode* is IMMEDIATEREPEAT, <mgmtCmd> shall be executed immediately and repeated multiple times as determined by *execNumber* and the time interval between each execution is specified by *execFrequency*. In this mode, *execDelay* shall not be used.
- If *execMode* is RANDOMONCE, <mgmtCmd> shall be executed only once at a delayed time which is specified by *execDelay*. In this mode, *execFrequency* and *execNumber* shall not be used.
- If *execMode* is RANDOMREPEAT, <mgmtCmd> shall be executed multiple times as specified by *execNumber* but the first execution shall be executed at a delayed time. *execDelay* specifies the delayed time. The time interval between each execution is specified by *execFrequency*.

7.4.16.2 <mgmtCmd> resource specific procedures for CRUD operations

7.4.16.2.0 Introduction

This clause describes <mgmtCmd> resource specific procedures for CRUD operations.

7.4.16.2.1 Create

This procedure shall use the Create common operations detailed in clause 7.3 without primitive-specific actions. The Originator shall use the steps described in clause 7.2.2.1. The Receiver shall use the steps described in clause 7.2.2.2.

The Originator shall provide the <mgmtCmd> resource representation to the Receiver (e.g. IN-CSE). The Receiver may generate one of the following status codes and send it to the Originator.

If the Originator provides an invalid *cmdType* value in the Create primitive, the Receiver shall generate a ***Response Status Code*** indicating "INVALID_CMDTYPE" error.

If the name/value entry in *execReqArgs* does not match the value of *cmdType* in the Create primitive, the Receiver shall generate a ***Response Status Code*** indicating "INVALID_ARGUMENTS" error.

If the name/value entries in *execReqArgs* do not contain mandatory arguments as required by *cmdType*, the Receiver shall generate a ***Response Status Code*** indicating "INSUFFICIENT_ARGUMENTS" error.

7.4.16.2.2 Retrieve

This procedure shall use the Retrieve common operations detailed in clause 7.3 without primitive specific actions. The Originator shall use the steps described in clause 7.2.2.1. The Receiver shall use the steps described in clause 7.2.2.2.

7.4.16.2.3 Update

7.4.16.2.3.1 Update (Normal)

If the Update primitive does not address the *execEnable* attribute of the <mgmtCmd> it results in update of all or part of the information of an existing <mgmtCmd> resource with the new attribute values. The procedure uses the common Update operations detailed in clause 7.3, without primitive specific actions.

The Originator shall use the steps described in clause 7.2.2.1. The Receiver shall use the steps described in clause 7.2.2.2.

If the Originator attempts to update attributes *execTarget*, *execMode*, but the <mgmtCmd> has a child resource <execInstance> already created, the Receiver shall generate a ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

If the Update primitive does address the *execEnable* attribute of the <mgmtCmd> it effectively triggers an Execute <mgmtCmd> procedure, see clause 7.4.16.2.3.2.

7.4.16.2.3.2 Update (Execute)

The execute operation is triggered by an Update primitive if the primitive addresses the *execEnable* attribute of the <mgmtCmd>. The procedure uses the Update common operations detailed in clause 7.2.2.2 with the following primitive specific operations after Recv-6.5:

1) The Receiver shall identify the managed entity and the management protocol. The *execTarget* attribute of <mgmtCmd> indicates the managed entity.

2) The Receiver shall automatically create an <execInstance> based on the <mgmtCmd> resource. If the *execTarget* attribute addresses a <group> resource, the Receiver shall create multiple <execInstance> sub-resources based on the value of *currentNrOfMembers* attribute.

3) The Receiver shall copy the following attributes from <mgmtCmd> to each <execInstance> created: *execMode*, *execFrequency*, *execDelay*, *execNumber*, and *execReqArgs*. The *execStatus* of <execInstance> is set to INITIATED. The Receiver shall set the *execTarget* attribute of each <execInstance> sub-resource to the URI of each target <node> resource.

4) The Receiver shall determine if the <mgmtCmd> shall be executed immediately or postponed according to the combination of *execMode*, *execFrequency*, *execDelay*, and *execNumber*. If the <mgmtCmd> shall be executed immediately (e.g. *execMode* is IMMEDIATEONCE), the following steps shall be performed; otherwise the following steps shall be postponed and skipped until the delay is expired (e.g. as indicated by *execDelay*).

5) The Receiver shall establish a management session with the identified managed entity.

6) The Receiver shall perform management command conversion and execution and set the *execStatus* attribute of <execInstance> to PENDING. If the Receiver cannot perform the command conversion successfully (e.g. *execReqArgs* does not have sufficient name/value pairs), the Receiver shall generate a ***Response Status Code*** indicating "MGMT_CONVERSION_ERROR" error.

7) After receiving completion response from the managed entity, the Receiver shall set *execStatus* attribute of corresponding <execInstance> to FINISHED.

If the Update primitive does not address the *execEnable* attribute of the <mgmtCmd>, it is treated as a normal Update, see clause 7.4.16.2.3.1.

7.4.16.2.4 Delete

This procedure is based on the Delete common operations detailed in clause 7.3.

The Receiver shall determine:

- If there are related management operations pending on the managed entity by checking if the *execStatus* attribute of all <execInstance> sub-resources are PENDING.

- If the related management operations are cancellable by checking the *cmdType* attribute of <mgmtCmd>.

If there are no management commands pending on the remote entity the Receiver shall delete the addressed <mgmtCmd> resource and send a success response to the Originator.

If there are cancellable management commands still pending on any remote entity, the Receiver shall perform the following steps:

1) The Receiver shall identify the managed entity and the management protocol. The *execTarget* attribute of each <execInstance> sub-resource which has *execStatus* of PENDING indicates the managed entity.

2) The Receiver shall establish a management session with each managed entity.

3) The Receiver shall perform management command conversion and execution resulting in cancellation of the commands which are pending on the managed entity.

4) For each successful cancellation RPC the *execStatus* attribute of the corresponding <execInstance> is set to CANCELLED. For each unsuccessful cancellation RPCs the *execStatus* attribute of the corresponding <execInstance> is determined from the reported fault codes for the unsuccessful RPCs.

5) Upon completion of all the cancellation operations, if any fault codes are returned by the managed entity, an error response to the Delete primitive with a ***Response Status Code*** indicating "CANCELLATION_FAILED" is returned, and the <mgmtCmd> resource is not deleted. If all cancellation operations are successful on the managed entity, a success response to the Delete primitive is returned and the <mgmtCmd> resource is deleted.

If there are non-cancellable management commands still pending on the remote entity, the Receiver shall send an error response to the Delete request to the Originator with a ***Response Status Code*** indicating "MGMT_COMMAND_NOT_CANCELLABLE" error. The *execStatus* attribute of the specific <execInstance> sub-resource is changed to STATUS_NON_CANCELLABLE.

7.4.17 Resource Type <execInstance>

7.4.17.1 Introduction

The <execInstance> resource shall contain the following child resource and attributes.

Table 7.4.17.1-1: Data type definition of <execInstance> resource

Data Type ID	File Name	Note
execInstance	CDT-execInstance-v3_11_0.xsd	

Table 7.4.17.1-2: Universal/Common Attributes of <execInstance> resource

Attribute Name	Request Optionality
	Update
@resourceName	NP
resourceType	NP
resourceID	NP
parentID	NP
accessControlPolicyIDs	O
creationTime	NP
expirationTime	O
lastModifiedTime	NP
labels	O
dynamicAuthorizationConsultationIDs	O

Table 7.4.17.1-3: Resource Specific Attributes of <execInstance> resource

Attribute Name	Request Optionality Update	Data Type	Default Value and Constraints
execStatus	NP	m2m:execStatusType	INITIATED, PENDING, FINISHED, CANCELLING, CANCELLED STATUS_NON_CANCELLABLE Default=INITIATED
execResult	NP	m2m:execResultType	No default
execDisable	O	xs:boolean	No default
execTarget	NP	m2m:nodeID	No default
execMode	NP	m2m:execModeType	IMMEDIATEONCE, IMMEDIATEREPEAT, RANDOMONCE, RANDOMREPEAT Default=IMMEDIATEONCE
execFrequency	NP	xs:duration	No default
execDelay	NP	xs:duration	Default=0
execNumber	NP	xs:nonNegativeInteger	Default=1
execReqArgs	NP	m2m:execReqArgsListType	No default

Table 7.4.17.1-4: Child Resources of <execInstance> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.17.2 <execInstance> resource specific procedures for CRUD operations

7.4.17.2.0 Create

The <execInstance> resource shall not be created via API.

7.4.17.2.1 Update (Cancel)

The <execInstance> Cancel operation is triggered by an Update primitive, if the primitive addresses the *execDisable* attribute. The procedure is based on Update common operations detailed in clause 7.3.

The Receiver shall determine:

- If there are related management operations pending on the managed entity by checking the *execStatus* attribute of the addressed <execInstance> sub-resource is PENDING.
- If the related management operations are cancellable by checking the *cmdType* attribute of the parent <mgmtCmd> resource.

If there are no management commands still pending on the remote entity, an error response to the Update primitive with a ***Response Status Code*** indicating "ALREADY_COMPLETE" is returned to the Originator.

If there are cancellable management commands still pending on the remote entity, the Receiver shall perform the following steps:

1) The Receiver shall identify the managed entity and the management protocol. The *execTarget* attribute of the addressed <execInstance> indicates the managed entity.
2) The Receiver shall establish a management session with the managed entity.
3) The Receiver shall perform management command conversion and execution resulting in cancellation of the commands which are pending on the managed entity.

4) If the cancellation is successfully executed on the managed entity, the Receiver shall return a success response to the Originator and shall set *execStatus* of <execInstance> to CANCELLED.

5) If the cancellation is unsuccessful on the managed entity, the Receiver shall return an error response to the Originator with a ***Response Status Code*** indicating "CANCELLATION_FAILED". The *execStatus* attribute is determined from the fault codes reported by the managed entity.

If there are non-cancellable management commands still pending on the remote entity, the Receiver shall return an error response to the Originator with a ***Response Status Code*** indicating "NOT_CANCELLABLE_COMMAND" and the *execStatus* attribute is changed to STATUS_NON_CANCELLABLE.

7.4.17.2.2 Retrieve

This procedure shall use the Retrieve common operations detailed in clause 7.3, without primitive specific actions. The Originator shall use the steps described in clause 7.2.2.1. The Receiver shall use the steps described in clause 7.2.2.2.

7.4.17.2.3 Delete

This procedure is based on the Delete common operations detailed in clause 7.3.

The Receiver shall determine:

- If there are related management operations pending on the managed entity by checking the *execStatus* attribute of the addressed <execInstance> sub-resource is PENDING.
- If the related management operations are cancellable by checking the *cmdType* attribute of the parent <mgmtCmd> resource.

If there are no management commands still pending on the remote entity, the Receiver shall delete the addressed resource and send a success response to the Originator.

If there are cancellable management commands still pending on the remote entity, the Receiver shall perform the following steps:

1) The Receiver shall identify the managed entity and the management protocol. The *execTarget* attribute of the addressed <execInstance> indicates the managed entity.

2) The Receiver shall establish a management session with the managed entity.

3) The Receiver shall perform management command conversion and execution resulting in cancellation of the commands which are pending on the managed entity.

4) If the cancellation is successfully executed on the managed entity, the Receiver shall return a success response to the Delete request to the Originator and shall delete the <execInstance> resource.

5) If the cancellation is unsuccessful on the managed entity, the Receiver shall return an error response to the Delete request to the Originator with a *Response Status Code* indicating "CANCELLATION_FAILED". The *execStatus* attribute is determined from the fault codes reported by the managed entity.

If there are non-cancellable management commands still pending on the remote entity, the Receiver shall return an error response to the Delete request to the Originator with a *Response Status Code* indicating "NOT_CANCELLABLE_COMMAND". The *execStatus* attribute is set to STATUS_NOT_CANCELLABLE.

7.4.18 Resource Type <node>

7.4.18.1 Introduction

The <node> resource represents specific information that provides properties of an oneM2M Node that can be utilized by other oneM2M operations. The <node> resource has <mgmtObj> as its child resources.

Table 7.4.18.1-1: Data type definition of <node> resource

Data Type ID	File Name	Note
node	CDT-node-v3_11_0.xsd	

Table 7.4.18.1-2: Universal/Common Attributes of <node> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.18.1-3: Resource Specific Attributes of <node> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
nodeID	M	O	m2m:nodeID	
hostedCSELink	O	O	m2m:ID	
hostedAELinks	O	O	m2m:listOfM2MID	
hostedServiceLinks	O	NP	m2m:listOfM2MID	
mgmtClientAddress	O	O	xs:string	
roamingStatus	NP	NP	xs:boolean	No default. true means that the Node is currently roaming. When this attribute is not present, it indicates that no information is available.
networkID	NP	NP	xs:string	No default. When this attribute is not present, it indicates that no information is available.

Table 7.4.18.1-4: Child resources of <node> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<mgmtObj>	[variable]	0..n	Clause 7.4.15, and Annex D
<subscription>	[variable]	0..n	Clause 7.4.8
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<transaction>	[variable]	0..n	Clause 7.4.61
<schedule>	[variable]	0..n	Clause 7.4.9

7.4.18.2 <node> resource specific procedures for CRUD operations

7.4.18.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.18.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.18.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.18.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.19 Resource Type <m2mServiceSubscriptionProfile>

7.4.19.1 Introduction

The <m2mServiceSubscriptionProfile> resource represents an M2M Service Subscription Profile. It is used to represent all data pertaining to the M2M Service Subscription Profile, i.e. the technical part of the contract between an M2M Application Service Provider and an M2M Service Provider.

The detailed description can be found in clause 9.6.19 in oneM2M TS-0001 [6].

Table 7.4.19.1-1: Data type definition of <m2mServiceSubscriptionProfile> resource

Data Type ID	File Name	Note
m2mServiceSubscriptionProfile	CDT-m2mServiceSubscriptionProfile-v3_11_0.xsd	

Table 7.4.19.1-2: Universal/Common Attributes of <m2mServiceSubscriptionProfile>

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
labels	O	O
lastModifiedTime	NP	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.19.1-3: Child resources of <m2mServiceSubscriptionProfile>

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<serviceSubscribedNode >	[variable]	0..n	Clause 7.4.20
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.19.2 <m2mServiceSubscriptionProfile> resource specific procedures for CRUD operations

7.4.19.2.0 Introduction

This clause describes <m2mServiceSubscriptionProfile> resource specific behaviour for CRUD operations.

7.4.19.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.19.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.19.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.19.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.20 Resource Type <serviceSubscribedNode>

7.4.20.1 Introduction

The <serviceSubscribedNode> resource represents M2M Node information that is needed as part of the M2M Service Subscription resource. It shall contain information about the M2M Node as well as application identifiers of the Applications running on that Node.

The detailed description can be found in clause 9.6.20 in oneM2M TS-0001 [6].

Table 7.4.20.1-1: Data type definition of <serviceSubscribedNode> resource

Data Type ID	File Name	Note
serviceSubscribedNode	CDT-serviceSubscribedNode-v3_11_0.xsd	

Table 7.4.20.1-2: Universal/Common Attributes of <serviceSubscribedNode> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
labels	O	O
lastModifiedTime	NP	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.20.1-3: Resource Specific Attributes of <serviceSubscribedNode> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
nodeID	M	NP	m2m:nodeID	
CSE-ID	O	NP	m2m:ID	
deviceIdentifier	O	NP	list of m2m:deviceID	
ruleLinks	O	O	list of xs:anyURI	
niddRequired	O	O	xs:boolean	No Default. If not configured, then IN-CSE default policy shall apply.

Table 7.4.20.1-4: Child resources of <serviceSubscribedNode> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.20.2 <serviceSubscribedNode> resource specific procedures for CRUD operations

7.4.20.2.0 Introduction

This clause describes <serviceSubscribedNode> resource specific behaviour for CRUD operations.

7.4.20.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.20.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.20.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.20.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.21 Resource Type <pollingChannel>

7.4.21.1 Introduction

The <pollingChannel> resource is used to perform service layer long polling when an AE/CSE cannot receive a request from other entities, however it can get a request as a response to a long polling request. Actual long polling can be performed on the <pollingChannelURI> resource which is the child resource of the <pollingChannel> resource.

The detailed description can be found in clause 9.6.21 in oneM2M TS-0001 [6].

Table 7.4.21.1-1: Data type definition of <pollingChannel> resource

Data Type ID	File Name	Note
pollingChannel	CDT-pollingChannel-v3_11_0.xsd	

Table 7.4.21.1-2: Universal/Common Attributes of <pollingChannel> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
expirationTime	O	O
labels	O	O

Table 7.4.21.1-3: Child resources of <pollingChannel> resource

Child Resource Type	Name	Multiplicity	Ref. to Resource Type Definition
<pollingChannelURI>	pcu	1	Clause 7.4.22
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.21.2 <pollingChannel> resource specific procedures for CRUD operations

7.4.21.2.0 Introduction

This clause describes <pollingChannel> resource specific behaviour for CRUD operations.

7.4.21.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except one addition:

- After Recv-6.3 procedure, the Hosting CSE shall check if the Originator ID is the same as the AE-ID or CSE-ID of the target <AE> resource or <remoteCSE> resource, respectively. If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

7.4.21.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except one addition:

- After Recv-6.3 procedure, the Hosting CSE shall check if the Originator ID is the same as the AE-ID or CSE-ID of the target <AE> resource or <remoteCSE> resource, respectively. If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

7.4.21.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except one addition:

- After Recv-6.3 procedure, the Hosting CSE shall check if the Originator ID is the same as the AE-ID or CSE-ID of the target <AE> resource or <remoteCSE> resource, respectively. If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

7.4.21.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except one addition:

- After Recv-6.3 procedure, the Hosting CSE shall check if the Originator ID is the same as the AE-ID or CSE-ID of the target <AE> resource or <remoteCSE> resource, respectively. If the check fails, then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

7.4.22 Resource Type <pollingChannelURI>

7.4.22.1 Introduction

The <pollingChannelURI> resource is the virtual child resource which is automatically generated during the parent <pollingChannel> resource creation. The detailed description can be found in clause 9.6.22 in oneM2M TS-0001 [6].

There is no data type definition for <pollingChannelURI> resource because it is a virtual resource type.

7.4.22.2 <pollingChannelURI> resource specific procedures for CRUD operations

7.4.22.2.0 Introduction

This clause describes <pollingChannelURI> resource specific behaviour for the Retrieve operation as a service layer long polling request. CUDN requests to the <pollingChannelURI> resource shall be rejected.

7.4.22.2.1 Create

The present document does not define Create operation on a <pollingChannelURI> resource. A Create request for the resource shall be rejected.

A <pollingChannelURI> virtual resource shall only be created during its parent <pollingChannel> resource creation procedure.

7.4.22.2.2 Retrieve

Originator:

Shall execute Originator actions in clause 7.2.2.1 as a service layer long polling request. It is the Originator's responsibility to initiate this procedure after it gets long polling response either successful or unsuccessful. The Originator shall send this Retrieve request as blocking request (clause 8.2.1 in oneM2M TS-0001 [6]).

Receiver:

Shall execute the following steps in order and these are modifications to the generic procedure from Recv-6.3 to Recv-6.5 in clause 7.2.2.2:

Recv-6.3 Check if the request Originator is the creator of the parent <pollingChannel> resource. If it is not the creator, the Hosting CSE shall send a response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

Recv-6.4 No change from the generic procedure.

Recv-6.5 If there is a pending request(s) to be sent to the Originator:

- Create a Response primitive by setting the ***Content*** parameter with one of the pending request(s).

Else:

- Wait for a request for the Originator until the ***Request Expiration Timestamp*** of the Originator's request. If a request is available before the ***Request Expiration Timestamp*** timeout, create a Response primitive setting the ***Content*** parameter to the received pending request. Otherwise, create a response primitive with a ***Response Status Code*** indicating "REQUEST_TIMEOUT" error.

7.4.22.2.3 Update

The present document does not define Update operation on a <pollingChannelURI> resource. An Update request for the resource shall be rejected.

7.4.22.2.4 Delete

The present document does not define Delete operation on a <pollingChannelURI> resource. A Delete request for the resource shall be rejected.

7.4.22.2.5 Notify

Request/response delivery mechanism via <pollingChannelURI> resource is depicted in the TS-0001 [6], Figure 10.2.5.12-1 (Request/response delivery via polling channel). In this procedure, the Originator is the Target AE/CSE and the Receiver is the <pollingChannelURI> Hosting CSE, respectively.

Originator:

This procedure follows the procedure specified in clause 7.2.2.1 with the following <pollingChannelURI> resource-specific updates.

Additional primitive specific operation on Orig-1.0:

- The Originator shall generate and populate a Notify request as described in clause 7.5.1.2.7.

Receiver:

The following are additional Hosting CSE procedures to the generic resource handling procedures from Recv-6.3 to Recv-6.5 in clause 7.2.2.2:

Recv-6.3 Check if the request Originator is the creator of the parent <pollingChannel> resource. If it is not the creator, the Hosting CSE shall send response primitive with a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error.

Recv-6.4 No change from the generic procedure.

Recv-6.5 Forward the response (step 006 in Figure 10.2.5.12-1 of TS-0001 [6]), which was contained in the ***Content*** parameter of the Notify request, to the entity that sent the associated request to the Hosting CSE (Originator in the figure 10.2.5.12-1). The associated request is the request that the Hosting CSE received and forwarded to the Registree AE or CSE over the polling channel (step 002 and step 004 in the figure). The association shall be done by matching the ***Request Identifier*** parameter of the request delivered in <pollingChannelURI> Retrieve response (step 004 in the figure) and the ***Request Identifier*** parameter of the response delivered in the ***Content*** parameter in a <pollingChannelURI> Notify request (step 005 in the figure).

7.4.23 Resource Type <statsConfig>

7.4.23.1 Introduction

The <statsConfig> resource is used to store configuration data for collecting statistics for AEs. The <eventConfig> child resource is a mechanism for defining events that trigger statistics collection activity. Additional description of the <statsConfig> resource is contained in clauses 9.6.23 and 10.2.11 of oneM2M TS-0001 [6].

Table 7.4.23.1-1: Data type definition of <statsConfig>

Data Type ID	File Name	Note
statsConfig	CDT-statsConfig-v3_11_0.xsd	

Table 7.4.23.1-2: Universal/Common Attributes of <statsConfig> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.23.1-3: Child resources of <statsConfig> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<eventConfig>	[variable]	0..n	Clause 7.4.24
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.23.2 <statsConfig> resource-specific procedures for CRUD operations

7.4.23.2.1 Create

Originator:

No change from the generic procedure in clause 7.2.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.2.2.2 with the following <statsConfig> resource-specific updates.

Resource-specific operation before Recv-6.2:

1) If the ***To*** primitive parameter addresses a receiver CSE that is not an IN-CSE, then the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

7.4.23.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.23.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.23.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.24 Resource Type <eventConfig>

7.4.24.1 Introduction

The <eventConfig> resource defines events that trigger statistics collection activity on an IN-CSE. Additional description of the <eventConfig> resource is contained in clauses 9.6.24 and 10.2.11 of oneM2M TS-0001 [6].

Table 7.4.24.1-1: Data type definition of <eventConfig>

Data Type ID	File Name	Note
eventConfig	CDT-eventConfig-v3_11_0.xsd	

Table 7.4.24.1-2: Universal/Common Attributes of <eventConfig> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.24.1-3: Resource Specific Attributes of <eventConfig> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
eventID	NP	NP	xs:string	Uniquely identifies a configurable event No default
eventType	M	O	m2m:eventType	DATAOPERATION STORAGEBASED TIMERBASED No default
eventStart	O	O	m2m:timestamp	No default (present only when *eventType* is set to TIMERBASED)
eventEnd	O	O	m2m:timestamp	No default (present only when *eventType* is set to TIMERBASED)
operationType	O	O	list of m2m:operation	CREATE RETRIEVE UPDATE DELETE NOTIFY No default (present only when *eventType* is set to DATAOPERATION)
dataSize	O	O	xs:nonNegativeInteger	No default (present only when *eventType* is set to STORAGEBASED)
eventResourceTypes	O	O	m2m:resourceTypeList	No default
eventResourceIDs	O	O	m2m:listOfURIs	No default

Table 7.4.24.1-4: Child Resources of <eventConfig> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.24.2 <eventConfig> resource-specific procedures for CRUD operations

7.4.24.2.1 Create

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.2.2.1, with the following <eventConfig> resource-specific updates.

Resource-specific operation before Orig-1.0 "Compose Request primitive":

1) If event-based statistics collection will be used, the Originator shall generate the representation of the <eventConfig> child resource instance to produce the desired trigger condition for the intended event. For example, one representation of <eventConfig> could have *eventType* set to DATA OPERATION and *operationType* set to Retrieve. In another example, a representation could have *eventType* set to TIMER-BASED, *eventStart* set to midnight tomorrow and *eventEnd* set to midnight of the day after tomorrow. See Table 7.4.24.1-3 for value restrictions and default settings pertaining to the attributes of <eventConfig>.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations.

1) If the *eventType* attribute is STORAGEBASED and the *dataSize* attribute is not specified in the Create request then the Hosting CSE shall reject the request with a BAD_REQUEST ***Response Status Code.***

2) If the Create request specifies both the *eventResourceTypes* attribute and the *eventResourceIDs* attribute then the Hosting CSE shall reject the request with a BAD_REQUEST ***Response Status Code.***

3) If the value of the *eventEnd* attribute is less than the *eventStart* attribute then the Hosting CSE shall reject the request with a BAD_REQUEST ***Response Status Code***.

No other change from the generic procedure in clause 7.2.2.2.

7.4.24.2.2 Retrieve

Originator:

No change from the generic procedure in clause 7.2.2.1.

Receiver:

No change from the generic procedure in clause 7.2.2.2.

7.4.24.2.3 Update

Originator:

No change from the generic procedure in clause 7.2.2.1.

Receiver:

Addition to the generic procedure in clause 7.2.2.2.

Recv-6.4: The following step is in addition to the procedures defined in clause 7.3.3.4

1) If the Update request would lead to both the *eventResourceTypes* attribute and the *eventResourceIDs* attributes being present in the resource, then the Hosting CSE shall reject the request with a BAD_REQUEST ***Response Status Code.***

7.4.24.2.4 Delete

Originator:

No change from the generic procedure in clause 7.2.2.1.

Receiver:

No change from the generic procedure in clause 7.2.2.2.

7.4.25 Resource Type <statsCollect>

7.4.25.1 Introduction

The <statsCollect> resource controls the collection of statistics information on an IN-CSE. Information in an associated <eventConfig> resource shall be used by the IN-CSE or IN-AE to define specific event-related triggers. Additional description of the <statsCollect> resource is contained in clauses 9.6.25 and 10.2.11 of oneM2M TS-0001 [6].

Table 7.4.25.1-1: Data type definition of <statsCollect>

Data Type ID	File Name	Note
statsCollect	CDT-statsCollect-v3_11_0.xsd	

Table 7.4.25.1-2: Universal/Common Attributes of <statsCollect> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.25.1-3: Resource Specific Attributes of <statsCollect> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
statsCollectID	NP	NP	xs:string	Unique ID (within SP domain) for each instance of collected statistics No default
collectingEntityID	M	NP	m2m:ID	Unique ID of entity (e.g. IN-AE, IN-CSE) requesting the collection of statistics No default
collectedEntityID	O	NP	m2m:listOfM2MID	List of Unique ID of entities (e.g. AE, CSE) for which statistics will be collected. If this attribute is not present in the resource, statistics shall be collected from all Entities
statsRuleStatus	M	O	m2m:statsRuleStatusType	ACTIVE INACTIVE No default
statModel	M	O	m2m:statModelType	EVENTBASED Default=EVENTBASED
collectPeriod	O	O	m2m:scheduleEntries	No default

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
				(see Table 7.4.9.1-3 for string format)
eventID	O	O	xs:string	Uniquely identifies a configurable event No default (present when *statModel* is set to EVENTBASED; corresponds to an *eventID* attribute in an <eventConfig> resource that defines a specific event for collection)

Table 7.4.25.1-4: Child Resources of <statsCollect> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.25.2 <statsCollect> resource-specific procedures for CRUD operations

7.4.25.2.1 Create

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.2.2.1, with the following <statsCollect> resource-specific updates.

Resource-specific operation before Orig-1.0:

1) The Originator shall generate and populate a representation of the <statsCollect> resource to produce the desired collection scenario, with the exception of statsCollectID (which is populated by the IN-CSE). If *statModel* is set to EVENT-BASED then the Originator shall provide a value for *eventID* that corresponds to an *eventID* value stored in an <eventConfig> resource (which defines the event triggers to be used).
See Table 7.4.25.1-3 for value restrictions and default settings pertaining to the attributes of <statsCollect>.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.2.2.2, with the following <statsCollect> resource-specific updates.

Resource-specific operation before Recv-6.2:

1) If the ***To*** primitive parameter addresses a receiver CSE that is not an IN-CSE, then the request shall be rejected with a ***Response Status Code*** indicating "BAD_REQUEST" error.

Resource-specific operations before Recv-6.6 and after Recv-6.5:

1) The receiver IN-CSE shall generate and store a unique (within the Service Provider domain) value for *statsCollectID*.

2) If *status* is set to ACTIVE, the IN-CSE shall begin monitoring the conditions defined by the <statsCollect> resource and generating Service Statistics Collection Records as the conditions are met.

7.4.25.2.2 Retrieve

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.2.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.2.2.2.

7.4.25.2.3 Update

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.2.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.2.2.2, with the following <statsCollect> resource-specific updates.

Resource-specific operation before Recv-6.6 and after Recv-6.5:

1) If *status* is set to ACTIVE, the IN-CSE shall begin monitoring the conditions defined by the <statsCollect> resource and generating Service Statistics Collection Records as the conditions are met.

2) If *status* is set to INACTIVE, the IN-CSE shall stop monitoring the conditions defined by the <statsCollect> resource.

7.4.25.2.4 Delete

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.2.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.2.2.2.

7.4.26 Announced resource types

7.4.26.1 Introduction

Instances of certain resource types can be announced by the Hosting CSE to one or more remote CSEs to inform the remote CSEs of the existence of the original resource instance. Table 7.4.26.1-1 lists the resource types which are announceable together with its resource type identifier and the name of the XSD file that specifies its data type.

Resource-specific attributes of a resource type may be declared announceable (mandatory or optional) or not announceable. Clause 9.6 in oneM2M TS-0001 [6] defines which attributes of each resource type are announceable.

The universal/common attributes of an announced resource type are listed in Table 7.4.26.1-2, together with their request optionality in Create and Update requests (see clause 7.4.1). The announced resource includes a common attribute *link* which represents a link to the original resource hosted by the original resource-Hosting CSE.

The child resources applicable to each announced resource type are listed in Table 9.6.26.1-1 of oneM2M TS-0001 [6].

Table 7.4.26.1-1: Data type definition of announced Resource types

Data Type ID	File Name	Note
accessControlPolicyAnnc	CDT-accessControlPolicy-v3_11_0.xsd	
remoteCSEAnnc	CDT-remoteCSE-v3_11_0.xsd	
AEAnnc	CDT-AE-v3_11_0.xsd	
containerAnnc	CDT-container-v3_11_0.xsd	
[flexContainer]Annc	See Annex J and oneM2M TS-0023 [40]	Announced variants of flexContainer specializations
contentInstanceAnnc	CDT-contentInstance-v3_11_0.xsd	
scheduleAnnc	CDT-schedule-v3_11_0.xsd	
locationPolicyAnnc	CDT-locationPolicy-v3_11_0.xsd	
groupAnnc	CDT-group-v3_11_0.xsd	
nodeAnnc	CDT-node-v3_11_0.xsd	
semanticDescriptorAnnc	CDT-semanticDescriptor-v3_11_0.xsd	
timeSeriesAnnc	CDT-timeSeries-v3_11_0.xsd	
timeSeriesInstanceAnnc	CDT-timeSeriesInstance-v3_11_0.xsd	
ontologyRepositoryAnnc	CDT-ontologyRepository-v3_11_0.xsd	
ontologyAnnc	CDT-ontology-v3_11_0.xsd	
semanticMashupJobProfileAnnc	CDT-semanticMashupJobProfile-v3_11_0.xsd	
semanticMashupInstanceAnnc	CDT-semanticMashupInstance-v3_11_0.xsd	
semanticMashupResultAnnc	CDT-semanticMashupResult-v3_11_0.xsd	
genericInterworkingServiceAnnc	CDT-genericInterworkingService-v3_11_0.xsd	
genericInterworkingOperationInstanceAnnc	CDT-genericInterworkingOperationInstance-v3_11_0.xsd	
svcObjWrapperAnnc	CDT-svcObjWrapper-v3_11_0.xsd	
svcFwWrapperAnnc	CDT-svcFwWrapper-v3_11_0.xsd	
allJoynAppAnnc	CDT-allJoynApp-v3_11_0.xsd	
allJoynSvcObjectAnnc	CDT-allJoynSvcObject-v3_11_0.xsd	
allJoynInterfaceAnnc	CDT-allJoynInterface-v3_11_0.xsd	
allJoynMethodAnnc	CDT-allJoynMethod-v3_11_0.xsd	
allJoynMethodCallAnnc	CDT-allJoynMethodCall-v3_11_0.xsd	
allJoynPropertyAnnc	CDT-allJoynProperty-v3_11_0.xsd	
multimediaSessionAnnc	CDT-multimediaSession-v3_11_0.xsd	

Table 7.4.26.1-2: Universal/Common Attributes of announcedResource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	M	O
lastModifiedTime	NP	NP
labels	O	O
link	M	O
dynamicAuthorizationConsultationIDs	O	O

Each announced resource type has the resource specific attributes that is the subset of the original resource.

Table 7.4.26.1-3: Resource Specific Attributes of announcedResource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
Name of attribute specified as MA	M	O	Same data type as defined for the original resource	This attribute shall be set to the same value with the attribute at the original resource
Name of attribute specified as OA	O	O	Same data type as defined for the original resource	This attribute shall be set to the same value with the attribute at the original resource

The procedures defined in the following clause 7.4.26.2 apply to all announced resources regardless of the resource type.

7.4.26.2 Resource specific procedures for CRUD operations

7.4.26.2.1 Introduction

This clause describes announced resource specific procedure for CRUD operations.

The original resource Hosting CSE shall create/update/delete the announced resource as specified at the clause 7.3.3.10 and clause 7.2.2.2.

7.4.26.2.2 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.26.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

In case of the ***Result Content*** information is set to the "original-resource", the Recv-6.5 shall be changed as follows:

Recv-6.5 "Read the original resource whose address is provided by the *link* attribute of the announced resource".

7.4.26.2.4 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2, except that Recv-6.3 is replaced as follows:

Recv-6.3 "Check if the value of the ***From*** parameter in Request message is identical to the CSE-ID included in the *link* attribute in the announced resource".

7.4.26.2.5 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2, except that Recv-6.3 is replaced as follows:

Recv-6.3 "Check if the value of the ***From*** parameter in Request message is identical to the CSE-ID included in the *link* attribute in the announced resource".

7.4.27 Resource Type latest

7.4.27.1 Introduction

The <latest> resource is a virtual resource because it does not have a representation. It is a child resource of the <container> and <timeSeries> resources. Whenever a request addresses the <latest> resource, the Hosting CSE shall apply the request to the latest <contentInstance> resource or the latest <timeSeriesInstance> resource among all existing <contentInstance> and <timeSeriesInstance> resources of the <container> resource or <timeSeries> resource.

7.4.27.2 <latest> Resource Specific Procedures for CRUD Operations

7.4.27.2.0 Introduction

This clause describes <latest> resource specific behaviour for operations. Only Retrieve and Delete operations shall be allowed for the <latest> resource.

7.4.27.2.1 Create

Originator:

The <latest> resource shall not be created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:
    a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.
2) "Send the Response primitive".

7.4.27.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

If the *locationID* of <container> is configured and the value of *locationUpdatePeriod* is marked '0' or not defined and *locationSource* attribute is 'Network Based', the following <*latest*> resource type specific procedures shall be performed after Recv-6.3 and before Recv-6.6 generic procedures.

1) The Hosting CSE shall transform the location-acquisition request into a Location Server request, using the attributes stored in <*locationPolicy*> resource. The Hosting CSE shall also provide default values for other required parameters (e.g. quality of position) in the Location Server request according to local policies.

The Hosting CSE shall send this Location Server request to the location server which could use either one of the two API interfaces: one is, OMA Mobile Location Protocol [i.4] and OMA RESTful NetAPI for Terminal Location [28]. The other one is 3GPP Monitoring Event API for terminal location [51]. The location server performs positioning procedure based upon the Location Server request.

2) The Hosting CSE shall receive the corresponding response and transform it into a Response primitive.

   a) If the positioning procedure is failed and *retrieveLastKnownLocation* is false, the Hosting CSE returns the response with a ***Response Status Code*** based on the error code and shall store a statusCode based on the error code in the *locationStatus* attribute in the <*locationPolicy*> resource. If the positioning procedure failed and *retrieveLastKnownLocation* is true and if the Hosting CSE supports the 3GPP API, the Hosting CSE shall repeat step 1), once only, but requesting the last known location from the Location Server using 3GPP Monitoring Event API [51].

   b) If the positioning procedure is successfully completed which means that the Hosting CSE acquires the location information, the Hosting CSE shall perform the procedures of creating a <*contentInstance*> as described in clause 7.4.7.2.1 according to the acquired location information and perform the normal Recv-6.5 procedure.

NOTE 1: For information on how the Location Server request message is converted to the OMA RESTful NetAPI for Terminal Location message, see clause G.2.

NOTE 2: For information on how the Location Server request message is converted to the 3GPP Monitoring Event API for terminal location message, see clause G.3.

If *locationID* of <container> is not configured, or the *locationID* of <container> is configured and the value of *locationUpdatePeriod* is greater than zero, or *locationSource* attribute is Device Based or Sharing Based:

1) No change from the generic procedures in clause 7.2.2.2 except the following modifications:

   - Recv-6.2 Check the existence of the latest <contentInstance> resource or the latest <timeSeriesInstance> resource among all existing <contentInstance> or <timeSeriesInstance> resources in the parent <container> or <timeSeries> resource. If the resource exists, the subsequent procedures of the Receiver (i.e. after Recv-6.2) shall be performed for the resource. If the resource does not exist, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

   - Recv-6.3 Addition to the normal procedure of Recv-6.3 "Check authorization of the Originator": Check the value of *disableRetrieval* attribute, and if the value is true then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

7.4.27.2.3 Update

Originator:

The <latest> resource shall not be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.27.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 except the following modification:

Recv-6.2 Check the existence of the latest <contentInstance> resource or the latest <timeSeriesInstance> resource among all existing <contentInstance> or <timeSeriesInstance> resources in the parent <container> or <timeSeries> resource. If the resource exists, the subsequent procedures of the Receiver (i.e. after Recv-6.2) shall be performed for the resource. If the resource does not exist, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

Recv-6.3 Addition to the normal procedure of Recv-6.3 "Check authorization of the Originator", Check the value of the *disableRetrieval* attribute, and if the value is true then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

7.4.28 Resource Type oldest

7.4.28.1 Introduction

The <oldest> resource is a virtual resource because it does not have a representation. It is a child resource of the <container> and <timeSeries> resources. Whenever a request addresses the <oldest> resource, the Hosting CSE shall apply the request to the oldest <contentInstance> resource or the oldest <timeSeriesInstance> resource among all existing <contentInstance> resources of the <container> resource or <timeSeries> resource.

7.4.28.2 <oldest> Resource Specific Procedures for CRUD Operations

7.4.28.2.0 Introduction

Only Retrieve and Delete operations shall be allowed for the <oldest> resource.

7.4.28.2.1 Create

Originator:

The <oldest> resource shall not be created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

    a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

    b) "Send the Response primitive".

7.4.28.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 except the following modification:

Recv-6.2 Check the existence of the oldest <contentInstance> resource or the oldest <timeSeriesInstance> resource among all existing <contentInstance> or <timeSeriesInstance> resources in the parent <container> or <timeSeries> resource. If the resource exists, the subsequent procedures of the Receiver (i.e. after Recv-6.2) shall be performed for the resource. If the resource does not exist, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

Recv-6.3 Addition to the normal procedure of Recv-6.3 "Check authorization of the Originator": Check the value of *disableRetrieval* attribute, and if the value is true then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

7.4.28.2.3 Update

Originator:

The <oldest> resource shall not be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the Response Status Code indicating an "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.28.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 except the following modification:

Recv-6.2 Check the existence of the oldest <contentInstance> resource or the oldest <timeSeriesInstance> resource among all existing <contentInstance> or <timeSeriesInstance> resources in the parent <container> or <timeSeries> resource. If the resource exists, the subsequent procedures of the Receiver (i.e. after Recv-6.2) shall be performed for the resource. If the resource does not exist, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating "NOT_FOUND" error.

Recv-6.3 Addition to the normal procedure of Recv-6.3 "Check authorization of the Originator": Check the value of *disableRetrieval* attribute, and if the value is true then the Hosting CSE shall reject the request with a ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

7.4.29 Resource Type <serviceSubscribedAppRule>

7.4.29.1 Introduction

The <serviceSubscribedAppRule> resource represents a rule that defines allowed App-ID and AE-ID combinations that are acceptable for registering an AE on a Registrar CSE. The detailed description can be found in the clause 9.6.29 in oneM2M TS-0001 [6].

Table 7.4.29.1-1: Data type definition of <serviceSubscribedAppRule> resource

Data Type ID	File Name	Note
serviceSubscribedAppRule	CDT-serviceSubscribedAppRule-v3_11_0.xsd	

Table 7.4.29.1-2: Universal/Common Attributes of <serviceSubscribedAppRule> resource

Attribute Name	Request Optionality	
	Create	Update
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.29.1-3: Resource Specific Attributes of <serviceSubscribedAppRule> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
applicableCredIDs	O	O	m2m:listOfM2MID	
allowedApp-IDs	O	O	m2m:listOfM2MID	
allowedAEs	O	O	m2m:listOfM2MID	
allowedRole-IDs	O	O	List of m2m:roleID	

Table 7.4.29.1-4: Child resources of <serviceSubscribedAppRule> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<*subscription*>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.29.2 <serviceSubscribedAppRule> resource specific procedures for CRUD operations

7.4.29.2.0 Introduction

This clause describes <serviceSubscribedAppRule> resource specific primitive behaviour for CRUD operations.

7.4.29.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.29.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.29.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.29.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.30 Resource Type <notificationTargetMgmtPolicyRef>

7.4.30.1 Introduction

This is a child resource of a <subscription> resource. This resource represents reference(s) to the policy to be followed by the Hosting CSE to manage Notification Targets of a resource subscription.

The detailed description can be found in clause 9.6.31 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.30.1-1: Data type definition of <notificationTargetMgmtPolicyRef> resource

Data Type ID	File Name	Note
notificationTargetMgmtPolicyRef	CDT-notificationTargetMgmtPolicyRef-v3_11_0.xsd	

Table 7.4.30.1-2: Universal/Common Attributes of <notificationTargetMgmtPolicyRef> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.30.1-3: Resource Specific Attributes of <notificationTargetMgmtPolicyRef> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
notificationTargetURI	M	O	list of xs:anyURI	No default
notificationPolicyID	O	O	xs:anyURI	

Table 7.4.30.1-4: Child resources of <notificationTargetMgmtPolicyRef> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.30.2 <notificationTargetMgmtPolicyRef> resource specific procedures for CRUD operations

7.4.30.2.0 Introduction

This clause describes <notificationTargetMgmtPolicyRef> resource specific primitive behaviour for CRUD operations.

7.4.30.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2, but with one additional check to in Recv-6.4 "Check validity of resource representation":

1) The Hosting CSE shall check if any Notification Target in the *notificationTargetURI* in the request primitive already exists in other <notificationTargetMgmtPolicyRef> resources which are the children of the targeted <subscription> resource. If so, the Hosting CSE shall send a response with ***Response Status Code*** indicating a "CONFLICT" error.

7.4.30.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.30.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.30.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.31 Resource Type <notificationTargetPolicy>

7.4.31.1 Introduction

This is a child resource of the <CSEBase> resource. This resource represents a policy to be followed by the Hosting CSE to manage the Notification Targets of a resource subscription. Each policy has Notification Target deletion rules.

The detailed description can be found in clause 9.6.32 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.31.1-1: Data type definition of <notificationTargetPolicy> resource

Data Type ID	File Name	Note
notificationTargetPolicy	CDT- notificationTargetPolicy -v3_11_0.xsd	

Table 7.4.31.1-2: Universal/Common Attributes of <notificationTargetPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.31.1-3: Resource Specific Attributes of <notificationTargetPolicy> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
action	O	NP	m2m:notificationTargetPolicyAction	No default
policyLabel	O	O	xs:token	No default
rulesRelationship	O	O	m2m:logicalOperator	This attribute shall have a value either "AND" or "OR" when more than one <*policyDeletionRules*> child resource is specified.

Table 7.4.31.1-4: Child resources of <notificationTargetPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<policyDeletionRules>	[variable]	0..2	Clause 7.4.32
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.31.2 <notificationTargetPolicy> resource specific procedures for CRUD operations

7.4.31.2.0 Introduction

This clause describes <notificationTargetPolicy> resource specific primitive behaviour for CRUD operations.

7.4.31.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.31.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.31.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.31.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.32 Resource Type <policyDeletionRules>

7.4.32.1 Introduction

This is a child resource of a <notificationTargetPolicy> resource. This resource represents rules to be applied by the Hosting CSE during Notification Target policy execution. Multiple rules are combined with AND or OR logical operation.

The detailed description can be found in clause 9.6.33 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.32.1-1: Data type definition of <policyDeletionRules> resource

Data Type ID	File Name	Note
policyDeletionRules	CDT- policyDeletionRules -v3_11_0.xsd	

Table 7.4.32.1-2: Universal/Common Attributes of <policyDeletionRules> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.32.1-3: Resource Specific Attributes of <policyDeletionRules> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
deletionRules	O	NP	m2m:deletionContexts	No default
deletionRulesRelation	O	O	m2m:logicalOperator	"OR" operation

Table 7.4.32.1-4: Child resources of <policyDeletionRules> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.32.2 <policyDeletionRules> resource specific procedures for CRUD operations

7.4.32.2.0 Introduction

This clause describes <policyDeletionRules> resource specific primitive behaviour for CRUD operations.

7.4.32.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2, but with one additional check in Recv-6.2 "Check existence of the addressed resource":

1) The Hosting CSE shall check if there are more than two existing <policyDeletionRules> resources as the children of the targeted <notificationTargetPolicy> resource. If so, the Hosting CSE shall send a response with ***Response Status Code*** indicating a "CONFLICT" error.

7.4.32.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.32.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.32.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.33 Resource Type <notificationTargetSelfReference>

7.4.33.1 Introduction

This is a child resource of a <subscription> resource. It is a virtual resource type so it has no resource representation nor XSD. Only the Delete operation is applicable to this resource.

7.4.33.2 <notificationTargetSelfReference> resource specific procedures for CRUD operations

7.4.33.2.0 Introduction

This clause describes <notificationTargetSelfReference> resource-specific primitive behaviour for CRUD operations.

7.4.33.2.1 Create

The <notificationTargetSelfReference> resource is not created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the Response Status Code indicating an "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.33.2.2 Retrieve

The <notificationTargetSelfReference> resource cannot be retrieved via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.33.2.3 Update

The <notificationTargetSelfReference> resource cannot be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating an "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.33.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 except the Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed". The Hosting CSE shall perform the Delete operation procedure for this resource type as follows:

1) Check if the Originator is listed as one of the Notification Targets in the *notificationTargetURI* attribute of a <*notificationTargetMgmtPolicyRef*> resource which is the sibling resource of the targeted <*notificationTargetSelfReference*> resource.
   If there is a resource, then the Hosting CSE shall fetch the <notificationTargetPolicy> resource linked from *notificationPolicyID* attribute.
   If there is no such resource, then the Hosting CSE shall fetch the default <notificationTargetPolicy> resource among children of the <CSEBase> resource as follows:

   a) The Hosting CSE shall find a <notificationTargetPolicy> resource which has the *policyLabel* attribute set as "Default" and the *creator* attribute set as the Originator ID.

   b) If there is no such resource in step a), then the Hosting CSE shall find the <notificationTargetPolicy> resource which has the *policyLabel* attribute set as "Default".
      Note that there is always the default <notificationTargetPolicy> resource created by the M2M Service Provider.

2) When there is one <policyDeletionRules> resource as a child of the fetched <notificationTargetPolicy> resource, the Hosting CSE evaluates the <policyDeletionRules> resource and perform action as specified in the *action* attribute.
   When there are two <policyDeletionRules> resources as a children of the fetched <notificationTargetPolicy> resource, the Hosting CSE evaluates the two <policyDeletionRules> resources and perform logical operation(i.e. AND or OR) as specified in the *rulesRelationship* attribute for the two evaluation results. If the logical operation result is true, then the Hosting CSE shall perform the action as specified in the *action* attribute.
   The action is performed as follows:

   a) If the action is the "accept request", the Hosting CSE shall remove the Originator in the *notificationURI* attribute in the <subscription> resource.

   b) If the action is the "reject request", the Hosting CSE shall send the response with ***Response Status Code*** indicating an "ORIGINATOR_HAS_NO_PRIVILEGE" error.

c) If the action is the "seek authorization from subscription originator before responding", the Hosting CSE shall send the Notify request with the information of the indication for Notification Target deletion, the Originator ID and the subscription reference from the Originator's request to the <subscription> resource creator (i.e. the *creator* attribute of the <subscription> resource). If the Hosting CSE gets the response from the creator with Notification Target removal allowance indication, then the Hosting CSE shall remove the Originator in the *notificationURI* attribute in the <subscription> resource.

d) If the action is the "inform the subscription originator without taking any action", the Hosting CSE shall send the Notify request with the information of the indication for Notification Target deletion, the Originator ID and the subscription reference from the Originator's request to the <subscription> resource creator (i.e. the *creator* attribute of the <subscription> resource) and send the response to the Originator with a ***Response Status Code*** indicating an "ORIGINATOR_HAS_NO_PRIVILEGE" error.

7.4.34 Resource Type <semanticDescriptor>

7.4.34.1 Introduction

The <semanticDescriptor> resource is used to store a semantic description pertaining to a resource and potentially sub-resources. Such a description shall be according to subject-predicate-object triples as defined in the RDF graph-based data model [34].

The detailed description can be found in clause 9.6.34 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.34.1-1: Data type definition of <semanticDescriptor> resource

Data Type ID	File Name	Note
semanticDescriptor	CDT- semanticDescriptor -v3_11_0.xsd	

Table 7.4.34.1-2: Universal/Common Attributes of <semanticDescriptor> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.34.1-3: Resource Specific Attributes of <semanticDescriptor> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
descriptorRepresentation	M	O	m2m:semanticFormat	No default
semanticOpExec	NP	O	m2m:sparql	No default
descriptor	M	O	xs:base64Binary	No default
ontologyRef	O	O	xs:anyURI	No default
relatedSemantics	O	O	list of xs:anyURI	No default
semanticValidated	NP	NP	xs:boolean	No default
validationEnable	O	O	xs:boolean	No default

Table 7.4.34.1-4: Child resources of <semanticDescriptor> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.34.2 <semanticDescriptor> resource specific procedures for CRUD operations

7.4.34.2.0 Introduction

This clause describes <semanticDescriptor> resource specific primitive behaviour for CRUD operations.

7.4.34.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) The Hosting CSE shall check that the *descriptor* attribute conforms to the syntax defined by the *descriptorRepresentation* attribute.

   b) If the *descriptor* attribute does not conform, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error .

   c) The Hosting CSE shall reject the request with a ***Response Status Code*** indicating a "BAD_REQUEST" error if the *descriptorRepresentation* attribute is set to "IRI".

2) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

   a) The Hosting CSE shall set the *validationEnable* attribute of the <semanticDescriptor> resource based on the value provided in the request and its local policy. Note that the local policy may override the suggested value provided in the request from the originator to enforce or disable the following semantic validation procedures. There are different cases depending on how the local policy is configured (which is out of the scope of the present document) and whether/how the *validationEnable* attribute is provided in the request:

      - *validationEnable* attribute is not present if it was not provided in the request or if the local policy does not allow for the *validationEnable* attribute;
      - *validationEnable* attribute is set to true or false according to the local policy no matter how the value is provided in the request;
      - *validationEnable* attribute is set to true or false according to the value provided in the request.

   b) If the *validationEnable* attribute is set as true, the hosting CSE shall perform the semantic validation process in the following steps according to clause 7.10.2 in oneM2M TS-0034 [50]. Otherwise, skip the following steps.

   c) Check if the addressed <*semanticDescriptor*> resource is linked to other <*semanticDescriptor*> resources on a remote CSE by the *relatedSemantics* attribute or by triples with annotation property *m2m:resourceDescriptorLink* in *descriptor* attribute. This process shall consider the recursive links.

- If yes, the Hosting CSE shall generate an Update request primitive with itself as the Originator and with the ***Content*** parameter set to the addressed <*semanticDescriptor*> resource representation, and send it to the <semanticValidation> virtual resource URI on the CSE which hosts the referenced ontology (following the *ontologyRef* attribute) of the addressed <*semanticDescriptor*> resource (see details in clause 7.4.48.2.3). After receiving the response primitive, i.e. the validation result, go to step k. If no response primitive was received due to time-out or other exceptional cases, the hosting CSE shall generate a ***Response Status Code*** indicating a "TARGET_NOT_REACHABLE" error.

- If no, perform the following steps.

d) Access the semantic triples from the *descriptor* attribute of the received <semanticDescriptor> resource.

e) Access the ontology referenced in the *ontologyRef* attribute of the received <semanticDescriptor> resource.

- If the ontology referenced by the *ontologyRef* attribute is an external ontology, not locally hosted by the Hosting CSE, the Hosting CSE shall retrieve it using the corresponding protocol and identifier information specified in the *ontologyRef* attribute.

- If the referenced ontology cannot be retrieved within a reasonable time (as defined by a local policy), the Hosting CSE shall generate a ***Response Status Code*** indicating an "ONTOLOGY_NOT_AVAILABLE" error.

f) Retrieve any local linked <semanticDescriptor> resources of the received <semanticDescriptor> resource following the URI(s) in the *relatedSemantics* attribute (if it exists) and the URI(s) in the triples with annotation property m2m:resourceDescriptorLink (if there are any).

- Repeat this step recursively to Retrieve any further local linked <semanticDescriptor> resources.

- If the local linked <semanticDescriptor> resources cannot be retrieved within a reasonable time (which is subject to a local policy), the Hosting CSE shall generate a ***Response Status Code*** indicating a "LINKED_SEMANTICS_NOT_AVAILABLE" error.

g) Retrieve the semantic triples from the *descriptor* attribute of the local linked <semanticDescriptor> resource.

h) Retrieve the referenced ontologies of the local linked <semanticDescriptor> resources following the URI(s) in *ontologyRef* attribute of the linked <semanticDescriptor> resources; If the referenced ontologies cannot be retrieved within a reasonable time (as defined by a local policy), the Hosting CSE shall generate a ***Response Status Code*** indicating an "ONTOLOGY_NOT_AVAILABLE" error.

i) Combine all the semantic triples of the addressed and local linked <semanticDescriptor> resources as the set of semantic triples to be validated, and combine all the referenced ontologies as the set of ontologies to validate the semantic triples against.

j) Check all the aspects of semantic validation according to clause 7.10.3 in oneM2M TS-0034 [50] based upon the semantic triples and referenced ontology. If any problem occurs, the Hosting CSE shall generate a ***Response Status Code*** indicating an "INVALID_SEMANTICS" error.

k) After the semantic validation process, the Hosting CSE shall set the *semanticValidated* attribute of the addressed <semanticDescriptor> resource according to the validation result (i.e. set to true if the no error occurs until now, otherwise false).

l) Based on its local policy, the Hosting CSE may also update the value of the *semanticValidated* attributes of the local linked <semanticDescriptor> resources according to the validation result.

7.4.34.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

- The *semanticOpExec* attribute is never returned in the response.

7.4.34.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

- The *descriptor* attribute can be updated using SPARQL as follows:

  Primitive specific operation on Orig-1.0 "Compose Request primitive": The originator creates a request to update the *semanticOpExec* attribute. The value of this attribute is set to a SPARQL request that includes INSERT, DELETE, or DELETE/INSERT with conditional SPARQL statements as defined in the SPARQL query language [33].

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) If both *semanticOpExec* and *descriptor* attributes exist, the Receiver shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

   b) If *semanticOpExec* attribute exists in the Request check that the syntax of its content corresponds to a valid SPARQL query request [33]. If the content does not correspond to a valid SPARQL query request, the Receiver shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

   c) If the *descriptor* attribute exists in the Request, check that the syntax of its content conforms to the syntax defined by the *descriptorRepresentation* attribute. If the content does not conform, the Receiver shall reject the request with a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

   d) The Hosting CSE shall reject the request with a ***Response Status Code*** indicating a "BAD_REQUEST" error if the *descriptorRepresentation* attribute is set to "IRI".

2) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" in addition:

   a) If *semanticOpExec* attribute exists in the Request, the Hosting CSE shall update the semantic triples in the *descriptor* attribute according to SPARQL update request in the *semanticOpExec* attribute. If the SPARQL update request cannot be executed, the Hosting CSE shall "create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "SPARQL_UPDATE_ERROR", otherwise proceed to step Recv-6.6.

   b) The hosting CSE shall set the *validationEnable* attribute of the addressed <semanticDescriptor> resource based on the value provided in the request and its local policy. Note that the local policy may override the suggested value provided in the request from the originator to enforce or disable the following semantic validation procedures. There are different cases depending on how the local policy is configured (which is out of the scope of the present document) and whether/how the *validationEnable* attribute is provided in the request:

      - no change to the existing *validationEnable* attribute if it is not provided in the request;
      - *validationEnable* attribute is not present if the local policy does not allow for the *validationEnable* attribute;
      - *validationEnable* attribute is set to true or false according to the local policy no matter how the value is provided in the request;
      - *validationEnable* attribute is set to true or false according to the value provided in the request.

   c) The hosting CSE shall perform steps 2b-2l as specified in clause 7.4.34.2.1.

d) If *validationEnable* attribute is changed from true to false, then the hosting CSE shall set the *semanticValidated* attribute of the addressed <semanticDescriptor> resource as false.

7.4.34.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.35 Resource Type <semanticFanOutPoint>

7.4.35.1 Introduction

The <*semanticFanOutPoint*> resource is a virtual resource because it does not have a representation; there are no common attributes, resource specific attributes or xsd. It is the child resource of a <group> resource. In the following descriptions, the general term *semantic resource* is used to refer to <*semanticDescriptor*> resources and any other future resources containing semantic information.

A <*semanticFanOutPoint*> can be addressed in one of two ways:

- Using the URI retrieved from its parent <group> resource; or
- Using a hierarchical URI formed by taking the hierarchical URI of the parent <group> and appending the string /sfop to that URI

Only Retrieve requests to the <*semanticFanOutPoint*> resource are valid, other operations are are rejected by the syntax check step at the Receiver. When a Retrieve request is received by the <*semanticFanOutPoint*> it triggers a semantic query operation if the ***Semantic Query Indicator*** parameter in the request primitive is set to true. Otherwise, it triggers a semantic resource discovery operation.

Semantic resource discovery is used to find resources in a CSE based on the semantic descriptions contained in the *descriptor* attribute of semantic resources. Since an overall semantic description (forming a graph [i.5]) may be distributed across a set of semantic resources, the semantic descriptions have to be retrieved (before or as needed) during the execution of the discovery request.

When using <semanticFanOutPoint>, the graph is distributed across the descriptors of the members of the parent <group> resource. The Group Hosting CSE indicates support for and availability of semantic functionality by setting the *semanticSupportIndicator* attribute of the <group> resource to true.

Similarly, semantic queries enable the retrieval of both explicitly and implicitly derived information based on syntactic, semantic and structural information contained in data (such as RDF data). The query result is produced by executing the semantic query statement over a set of distributed semantic resources, which are the members of the parent <group> resource and they constitute the RDF data basis (forming a graph) for the query statement to be executed on.

Targeting the <semanticFanOutPoint> virtual resource results in creating and distributing retrieve operations to all the member resources. The results of the retrieve requests are used to form an aggregated semantic description containing the overall graph, on which the graph pattern matching is performed for supporting semantic query or semantic resource discovery operations.

7.4.35.2 <semanticFanOutPoint> resource specific procedures for CRUD operations

7.4.35.2.0 Introduction

This clause describes <semanticFanOutPoint> resource specific primitive behaviour for CRUD operations.

7.4.35.2.1 Create

Originator:

The <semanticFanOutPoint> resource shall not support Create operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.35.2.2 Retrieve

Originator:

No primitive specific operations.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-6.2 specified in clause 7.2.2.2 Generic Resource Request Procedure for Receiver, with the following primitive specific operations:

After Recv-1.0 "Check the validity of received request primitive":

1) Check that the syntax of the ***semanticsFilter*** corresponds to a valid SPARQL query request [33]. If the ***semanticsFilter*** does not correspond to a valid SPARQL query request, the Receiver shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

2) If the ***Semantic Query Indicator*** parameter included in the request message is set to true, the request shall be processed as a semantic query. Otherwise, the request shall be processed as a semantic resource discovery.

After Recv-6.2 "Check existence of the addressed resource":

1) Check that the *semanticSupportIndicator* of the parent <group> resource is set to true.

2) Check the authorization of the Originator using the *membersAccessControlPolicyIDs* of the parent group resource. In the case the *membersAccessControlPolicyIDs* is not provided, the *accessControlPolicyIDs* of the parent group resource shall be used.

3) Fan-out <semanticDescriptor> Retrieve Requests to each CSE hosting sub-groups or members as follows:

   For each group member, the Hosting CSE shall perform the following steps:

   a) The primitive parameters ***From*** and ***To*** shall be mapped to corresponding Retrieve Requests to be sent out to each member of the group. The primitive parameter ***From*** shall be used directly. The prefix of primitive parameter ***To*** i.e. <URI of group resource>/sfop shall be replaced by hierarchical URIs derived from the attribute *memberIDs* of the <group> resource.

   b) The group hosting CSE shall execute "Compose Request primitives" with the ***semanticsFilter*** filter condition set to false.

   c) "Send the Request to the receiver CSE".

   d) "Wait for Response primitive".

4) Once the Responses to the Retrieve Requests have been received, proceed to the following steps:

   a) Aggregate the descriptors from the Retrieve Responses and deliver the content for SPARQL processing, along with the ***semanticsFilter*** content.

   b) Wait for a SPARQL processing response.

c) Perform Recv-6.7 "Create a success response" where the Response shall include the SPARQL processing result. In case of semantic query, the Response shall include the semantic query result. In case of semantic resource discovery, the Response shall include a list of identified resource URIs.

d) Perform Recv-6.8 and the procedure is terminated.

7.4.35.2.3 Update

Originator:

The <semanticFanOutPoint> resource shall not support Update operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.35.2.4 Delete

Originator:

The <semanticFanOutPoint> resource shall not support Delete operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.36 Resource Type <dynamicAuthorizationConsultation>

7.4.36.1 Introduction

A <dynamicAuthorizationConsultation> resource shall be used by a CSE to perform consultation-based dynamic access control to resources as specified in the present document and in oneM2M TS-0003 [7].

The detailed description can be found in clause 9.6.43 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.36.1-1: Data type definition of <dynamicAuthorizationConsultation> resource

Data Type ID	File Name	Note
dynamicAuthorizationConsultation	CDT-dynamicAuthorizationConsultation-v3_11_0.xsd	

Table 7.4.36.1-2: Universal/Common Attributes of <dynamicAuthorizationConsultation > resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicyIDs	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.36.1-3: Resource Specific Attributes of <dynamicAuthorizationConsultation> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
dynamicAuthorizationEnabled	M	O	xs:boolean	No default
dynamicAuthorizationPoA	M	O	m2m:poaList	No default
dynamicAuthorizationLifetime	O	O	m2m:timestamp	No default

Table 7.4.36.1-4: Child resources of <dynamicAuthorizationConsultation> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.36.2 <dynamicAuthorizationConsultation> resource specific procedures for CRUD operations

7.4.36.2.0 Introduction

This clause describes <dynamicAuthorizationConsultation> resource specific primitive behaviour for CRUD operations.

7.4.36.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.36.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.36.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.36.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.37 Resource Type <flexContainer>

7.4.37.1 Introduction

This resource represents a customizable container for data instances. It is a template for the definition of flexible specializations of data containers.

The detailed description can be found in clause 9.6.35 in oneM2M TS-0001 [6].

There are multiple specializations of <flexContainer> specified by oneM2M. Each of these specializations has its own schema file. There is no separate schema file just for <flexContainer>, however the XML schema types for the specializations all conform to the pattern described in this clause. The XSD of <flexContainer> specializations may use a targetNamespace other than the one identified by the *m2m:* prefix. Specializations of <flexContainer> which employ the namespace prefix *m2m:* are defined in Annex J. Specialization of <flexContainer> which employ the namespace prefix *hd:* for *Home Domain* use cases are specified in oneM2M TS-0023 [40].

The following resource types are allowed to include <flexContainer> specializations as children: <CSEBase>, <AE>, <remoteCSE> and <container>.

Table 7.4.37.1-1: Universal/Common Attributes of <flexContainer> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
stateTag	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
creator	O	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.37.1-2: Resource Specific Attributes of <flexContainer> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
contentSize	NP	NP	xs:nonNegativeInteger	No default
nodeLink	O	O	xs:anyURI	No default
[customAttribute]	O	O	Name and data type are defined in the specification document or XSD file identified by the value of containerDefinition attribute.	No default

Table 7.4.37.1-3: Child Resources of <flexContainer> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<subscription>	[variable]	0..n	Clause 7.4.8
<container>	[variable]	0..n	Clause 7.4.6
<flexContainer>	[variable]	0..n	Clause 7.4.37
<timeSeries>	[variable]	0..n	Clause 7.4.38
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.37.2 <flexContainer> resource specific procedures for CRUD operations

7.4.37.2.0 Introduction

This clause describes <flexContainer> resource specific behaviour for CRUD operations.

7.4.37.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 with the following additional operations.

1) The hosting CSE shall validate the received resource representation against the schema value present in the received resource *containerDefinition* attribute. If the schema is not available then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "SPECIALIZATION_SCHEMA_NOT_FOUND" error. If the received resource is not valid then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "BAD_REQUEST" error.

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations:

1) The Hosting CSE shall set the *contentSize* attribute to the sum of the size in bytes of all of the custom attributes.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.37.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.37.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 with the following additional operations.

1) The hosting CSE shall validate the received resource representation against the schema value present in the resource *containerDefinition* attribute. If the received resource is not valid then the Hosting CSE shall return a response primitive with a ***Response Status Code*** indicating "BAD_REQUEST" error.

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations:

1) The Hosting CSE shall update the *contentSize* attribute to the sum of the size in bytes of all of the custom attributes.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.37.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.38 Resource Type <timeSeries>

7.4.38.1 Introduction

The resource represents a container for Time Series Data instances. It is used to share information with other entities and potentially to track, detect and report the missing data in Time Series. A <*timeSeries*> resource has no associated content, only attributes and child resources.

The detailed description can be found in clause 9.6.36 in oneM2M TS-0001 [6].

Table 7.4.38.1-1: Data type definition of <timeSeries> resource

Data Type ID	File Name	Note
timeSeries	CDT-timeSeries-v3_11_0.xsd	

Table 7.4.38.1-2: Universal/Common Attributes of <timeSeries> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.38.1-3: Resource Specific Attributes of <timeSeries> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
maxNrOfInstances	O	O	xs:nonNegativeInteger	No default
maxByteSize	O	O	xs:nonNegativeInteger	No default
maxInstanceAge	O	O	xs:nonNegativeInteger	No default
currentNrOfInstances	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the *maxNrOfInstances*)
currentByteSize	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the *maxByteSize*)
periodicInterval	O	O	xs:positiveInteger	No default (This is in units of milliseconds)
missingDataDetect	O	O	xs:boolean	No default
missingDataMaxNr	O	O	xs:positiveInteger	No default
missingDataList	NP	NP	m2m:missingDataList	No default
missingDataCurrentNr	NP	NP	xs:nonNegativeInteger	No default (This is generated by the Hosting CSE and limited by the *missingDataMaxNr*)
missingDataDetectTimer	O	O	xs:positiveInteger	No default (This is in units of milliseconds)
ontologyRef	O	O	xs:anyURI	No default
contentInfo	O	O	m2m:contentInfo	No default

Table 7.4.38.1-4: Child Resources of <timeSeries> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<timeSeriesInstance>	[variable]	0..n	Clause 7.4.39
<subscription>	[variable]	0..n	Clause 7.4.8
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<latest>	la	1	Clause 7.4.27
<oldest>	ol	1	Clause 7.4.28
<transaction>	[variable]	0..n	Clause 7.4.61
NOTE: <latest> and <oldest> are only present if there is a <timeSeriesInstance>.			

7.4.38.2 <timeSeries> resource specific procedures for CRUD operations

7.4.38.2.0 Introduction

This clause describes <timeSeries> resource specific behaviour for CRUD operations.

7.4.38.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation after Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed ". See clause 7.2.2.2.

In the case that the *periodicInterval* attribute is set and *missingDataDetect* is true, the Hosting CSE shall monitor the Time Series Data based on its *periodicInterval*. The Hosting CSE shall consider the expecting Time Series Data be lost when the amount of time equal to *missingDataDetectTimer* has passed relative to its expecting generation time when the data was generated by the AE/CSE.

When the Hosting CSE detects a missing data point, the *dataGenerationTime* of the missing data point is inserted into the *missingDataList* attribute and the *missingDataCurrentNr* shall be increased by one. When the *missingDataCurrentNr* reaches the *missingDataMaxNr*, the oldest *dataGenerationTime* shall be removed from *missingDataList* to enable the insertion of the new missing data point information.

7.4.38.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.38.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.38.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.39 Resource Type <timeSeriesInstance>

7.4.39.1 Introduction

The <*timeSeriesInstance*> resource represents a data instance in the <*timeSeries*> resource.

The detailed description can be found in clause 9.6.37 in oneM2M TS-0001 [6].

Table 7.4.39.1-1: Data type definition of <timeSeriesInstance> resource

Data Type ID	File Name	Note
timeSeriesInstance	CDT-timeSeriesInstance-v3_11_0.xsd	

Table 7.4.39.1-2: Universal/Common Attributes of <timeSeriesInstance> resource

Attribute Name	Request Optionality
	Create
@resourceName	O
resourceType	NP
resourceID	NP
parentID	NP
creationTime	NP
expirationTime	O
lastModifiedTime	NP
labels	O
announceTo	O
announcedAttribute	O

Table 7.4.39.1-3: Resource Specific Attributes of <timeSeriesInstance> resource

Attribute Name	Request Optionality	Data Type	Default Value and Constraints
	Create		
dataGenerationTime	M	m2m:absRelTimestamp	No default
content	M	xs:anySimpleType	No default
sequenceNr	O	xs:nonNegativeInteger	No default
contentSize	NP	xs:nonNegativeInteger	No default

Table 7.4.39.1-4: Child resources of <timeSeriesInstance> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<semanticDescriptor>	[variable]	0..n	Clause 7.4.34
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.39.2 <timeSeriesInstance> resource specific procedures for CRUD operations

7.4.39.2.0 Introduction

This clause describes <timeSeriesInstance> resource specific behaviour for CRUD operations.

7.4.39.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

The Originator shall maintain an internal counter to generate *sequenceNr* which is increased by one. When the *sequenceNr* reaches to the *maxNrOfInstances* of the direct parent <timeSeries> resource, it shall be set to one.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) Steps for Create procedure of <timeSeriesInstance> resource shall be same as that steps of <contentInstance> resource described in clause 7.4.7.2.1, except <container> resource in that procedure would correspond to <timeSeries> resource and <contentInstance> resource would correspond to <timeSeriesInstance> resource.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.39.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.39.2.3 Update

Originator:

The <timeSeriesInstance> resource shall not be Updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.39.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

1) *currentNrOfInstances* and *currentByteSize* of direct parent <timeSeries> resource shall be updated.

7.4.40 Resource Type <role>

7.4.40.1 Introduction

The <role> resource represents a role that is assigned to an AE or CSE.

The detailed description can be found in clause 9.6.38 in oneM2M TS-0001 [6].

Table 7.4.40.1-1: Data type definition of <role> resource

Data Type ID	File Name	Note
role	CDT-role-v3_11_0.xsd	

Table 7.4.40.1-2: Universal/Common Attributes of <role> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
expirationTime	O	O
labels	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.40.1-3: Resource Specific Attributes of <role> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
roleID	M	NP	m2m:roleID	No default
issuer	M	NP	m2m:ID	No default
holder	M	NP	m2m:ID	No default
notBefore	M	NP	m2m:timestamp	No default
notAfter	M	NP	m2m:timestamp	No default
roleName	O	NP	xs:string	No default
tokenLink	O	O	xs:anyURI	No default

Table 7.4.40.1-4: Child Resources of <role> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.40.2 <role> resource specific procedures for CRUD operations

7.4.40.2.0 Introduction

This clause describes <role> resource specific behaviour for CRUD operations.

7.4.40.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.40.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.40.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.40.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.41 Resource Type <token>

7.4.41.1 Introduction

The <token> resource represents a token that is issued to an AE or CSE.

The detailed description can be found in clause 9.6.39 in oneM2M TS-0001 [6].

Table 7.4.41.1-1: Data type definition of <token> resource

Data Type ID	File Name	Note
token	CDT-token-v3_11_0.xsd	

Table 7.4.41.1-2: Universal/Common Attributes of <token> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
expirationTime	O	O
labels	O	O
creationTime	NP	NP
lastModifiedTime	NP	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.41.1-3: Resource Specific Attributes of <token> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
tokenID	M	NP	m2m:tokenID	No default
tokenObject	M	NP	m2m:dynAuthJWT	No default
version	O	NP	xs:string	No default
issuer	O	NP	m2m:ID	No default
holder	O	NP	m2m:ID	No default
notBefore	O	NP	m2m:timestamp	No default
notAfter	O	NP	m2m:timestamp	No default
tokenName	O	NP	xs:string	No default
audience	O	NP	list of m2m:ID	No default
permissions	O	NP	m2m:tokenPermissions	No default
extension	O	NP	xs:string	No default

Table 7.4.41.1-4: Child Resources of <token> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.41.2 <token> resource specific procedures for CRUD operations

7.4.41.2.0 Introduction

This clause describes <token> resource specific behaviour for CRUD operations.

7.4.41.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.41.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.41.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.41.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.42 Void

7.4.43 Resource Type <authorizationDecision>

7.4.43.1 Introduction

The <authorizationDecision> resource represents an access control decision. The detailed description can be found in clause 9.6.41 in oneM2M TS-0001 [6].

Table 7.4.43.1-1: Data type definition of <authorizationDecision> resource

Data Type ID	File Name	Note
authorizationDecision	CDT-authorizationDecision-v3_11_0.xsd	

Table 7.4.43.1-2: Universal/Common Attributes of <authorizationDecision> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.43.1-3: Resource Specific Attributes of <authorizationDecision> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
decision	NP	NP	m2m:authorizationDecision	No default
status	NP	NP	m2m:authorizationStatus	No default
to	O	O	xs:anyURI	No default
from	O	O	m2m:ID	No default
operation	O	O	m2m:operation	No default
requestedResourceType	O	O	m2m:resourceType	No default
filterUsage	O	O	m2m:filterCriteria	No default
roleIDs	O	O	list of m2m:roleID	No default
tokenIDs	O	O	list of m2m:tokenID	No default
tokens	O	O	list of m2m:dynAuthJWT	No default
requestTime	O	O	m2m:timestamp	No default
originatorLocation	O	O	m2m:locationRegion	No default
originatorIP	O	O	m2m:ipAddress	No default

Table 7.4.43.1-4: Child Resources of <authorizationDecision> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.43.2 <authorizationDecision> resource specific procedures for CRUD operations

7.4.43.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.43.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.43.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

The Hosting CSE shall check whether a PDP procedure is bound to the <authorizationDecision> resource. If it is, the Hosting CSE shall trigger the PDP procedure, otherwise no other changes from the generic procedures in clause 7.2.2.2.

The triggered PDP procedure shall perform as follows:

1) Check if the access control decision request satisfies the following conditions. If it does, continue with the remaining steps, otherwise stop this procedure and return an error ***Response Status Code*** of "BAD_REQUEST":

   a) Only *to*, *from*, *operation*, *resourceType*, *filterUsage*, *roleIDs*, *tokenIDs*, *tokens*, *requestTime*, *originatorLocation*, and/or *originatorIP* resource attributes may be in the update request.

   b) All the mandatory resource attributes (i.e. *to*, *from* and *operation* attributes) used for constructing an access control decision request are in the update request.

   c) The formats of the updated values are all correct.

2) Delete any existing resource specific attributes before performing the resource update operation, and then perform the resource update operation.

3) Construct an access control decision request with the updated resource attributes.

4) Obtain applicable access control policies and information according to the access control decision request. If it is not successful, the *status* attribute shall be set as follows:

a) If the procedure cannot obtain applicable access control policies or required access control information without error, update the *status* attribute with "NOT_APPLICABLE" and go to step 6.

b) If the procedure cannot obtain applicable access control policies or required access control information with error, update the *status* attribute with "INDETERMINATE" and go to step 6.

NOTE: How to obtain the applicable access control policies or required access control information is out of scope of the present document.

5) Evaluate the access control decision request against access control policies as specified in oneM2M TS-0003 [7] and update the *decision* and *status* attributes with the evaluation result as follows:

a) If some access control information required by the evaluation procedure is not provided by the request, update the *status* attribute with "MISSING_ATTRIBUTE" and go to step 6.

b) If there are some errors in access control policies and/or tokens, update the *status* attribute with "SYNTAX_ERROR" and go to step 6.

c) If an error occurs during the evaluation process, update the *status* attribute with "PROCESSING_ERROR" and go to step 6.

d) If the access control policy evaluation is successful, update the *decision* attribute with the evaluation result (i.e. "PERMIT" or "DENY") and the *status* attribute with "OK" and go to step 6.

6) Generate an UPDATE response using the *decision* and *status* attributes and returning it back to the requester.

7) Delete all the resource specific attributes after the response has been sent in order to avoid an information leak.

7.4.43.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.44 Resource Type <authorizationPolicy>

7.4.44.1 Introduction

The <authorizationPolicy> resource represents a set of access control policies. The detailed description can be found in clause 9.6.42 in oneM2M TS-0001 [6].

Table 7.4.44.1-1: Data type definition of <authorizationPolicy> resource

Data Type ID	File Name	Note
authorizationPolicy	CDT-authorizationPolicy-v3_11_0.xsd	

Table 7.4.44.1-2: Universal/Common Attributes of <authorizationPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.44.1-3: Resource Specific Attributes of <authorizationPolicy> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
policies	NP	NP	m2m:setOfPermissions	No default
combiningAlgorithm	NP	NP	m2m:acpCombiningAlgorithm	No default
status	NP	NP	m2m:authorizationStatus	No default
to	O	O	xs:anyURI	No default

Table 7.4.44.1-4: Child Resources of <authorizationPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.44.2 <authorizationPolicy> resource specific procedures for CRUD operations

7.4.44.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.44.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.44.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

The Hosting CSE shall check whether a PRP procedure is bound to the <authorizationPolicy> resource. If it is, the Hosting CSE shall trigger the PRP procedure, otherwise no other changes from the generic procedures in clause 7.2.2.2.

The triggered PRP procedure shall perform as follows:

1) Check if the access control policy request satisfies the following conditions. If it does, continue with the remaining steps, otherwise stop this procedure and return an error ***Response Status Code*** of "BAD_REQUEST":

   a) Only the *to* resource attribute is in the update request.

   b) The format of the updated value is correct.

2) Delete any existing resource specific attributes before performing the resource update operation, and then perform the resource update operation.

3) Construct an access control policy request with the updated resource attributes.

4) Obtain applicable access control policies and policy combining algorithm according to the access control policy request and set the *policies*, *combiningAlgorithm* and *status* attributes as follows:

   a) If the procedure cannot obtain applicable access control policies without error, update the *status* attribute with "NOT_APPLICABLE" and go to step 5.

   b) If an error occurs during the access control policy retrieval process, update the *status* attribute with "PROCESSING_ERROR" and go to step 5.

   c) If the access control policy retrieval is successful, update the *policies* attribute with the retrieved access control policies, *combiningAlgorithm* attribute with the retrieved policy combining algorithm and *status* attribute with "OK" and go to step 5.

NOTE: How to obtain the applicable access control policies and policy combining algorithm is out of scope of the present document.

5) Generate an UPDATE response using the *policies*, *combiningAlgorithm* and *status* attributes and return it to the requester.

6) Delete all the resource specific attributes after the response has been sent in order to avoid an information leak.

7.4.44.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.45 Resource Type <authorizationInformation>

7.4.45.1 Introduction

The <authorizationInformation> resource represents access control information. The detailed description can be found in clause 9.6.43 in oneM2M TS-0001 [6].

Table 7.4.45.1-1: Data type definition of <authorizationInformation> resource

Data Type ID	File Name	Note
authorizationInformation	CDT-authorizationInformation-v3_11_0.xsd	

Table 7.4.45.1-2: Universal/Common Attributes of <authorizationInformation> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.45.1-3: Resource Specific Attributes of <authorizationInformation> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
status	NP	NP	m2m:authorizationStatus	No default
from	O	O	m2m:ID	No default
roleIDs	O	O	List of m2m:roleID	No default
tokenIDs	O	O	List of m2m:tokenID	No default

Table 7.4.45.1-4: Child Resources of <authorizationInformation> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<role>	[variable]	0..n	Clause 7.4.40
<token>	[variable]	0..n	Clause 7.4.41
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.45.2 <authorizationInformation> resource specific procedures for CRUD operations

7.4.45.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.45.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.45.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

The Hosting CSE shall check whether a PIP procedure is bound to the <authorizationInformation> resource. If it is, the Hosting CSE shall trigger the PIP procedure, otherwise no other changes from the generic procedures in clause 7.2.2.2.

The triggered PIP procedure shall perform as follows:

1) Check if the access control information request satisfies the following conditions. If it does, continue with the remaining steps, otherwise stop this procedure and return an error ***Response Status Code*** of "BAD_REQUEST":

   a) Only *from*, *roleIDs* and/or *tokenIDs* resource attributes may be in the update request.

   b) All the mandatory resource attributes (i.e. *from* attribute) used for constructing an access control information request are in the update request.

   c) The format of the updated value is correct.

2) Delete any existing resource specific attributes and child resources before performing the resource update operation, and then perform the resource update operation.

3) Construct an access control information request with the updated resource attributes.

4) Obtain requested access control information according to the access control information request and create <role> and/or <token> child resources and update the *status* attribute as follows:

   a) If the procedure cannot obtain required access control information without error, update the *status* attribute with "NOT_APPLICABLE" and go to step 5.

   b) If an error occurs during the access control information retrieval process, update the *status* attribute with "PROCESSING_ERROR" and go to step 5.

   c) If the access control information retrieval is successful, create corresponding <role> and/or <token> child resources and update the *status* attribute with "OK" and go to step 5.

NOTE: How to obtain the requested access control information is out of scope of the present document.

5) Generate an UPDATE response using the <role> and/or <token> resources and *status* attribute and return it to the requester.

6) Delete all the resource specific child resources and attributes after the response has been sent in order to avoid an information leak.

7.4.45.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.46 Resource Type <*ontologyRepository*>

7.4.46.1 Introduction

The <ontologyRepository> resource is a repository of ontologies used for reference and management of the associated <ontology> resources. It also provides semantic validation functionality as detailed in clause 6.7 in oneM2M TS-0034 Semantics Support [50].

The detailed description can be found in clause 9.6.50 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.46.1-1: Data type definition of <*ontologyRepository*> resource

Data Type ID	File Name	Note
ontologyRepository	CDT-ontologyRepository-v3_11_0.xsd	

Table 7.4.46.1-2: Universal/Common Attributes of <*ontologyRepository*> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

There is no resource specific attributes of <*ontologyRepository*> resource.

Table 7.4.46.1-3: Child Resources of <*ontologyRepository*> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<*ontology*>	[variable]	0..n	Clause 7.4.47
<*semanticValidation*>	smv	1	Clause 7.4.48
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.46.2 <ontologyRepository> resource specific procedures for CRUD operations

7.4.46.2.0 Introduction

This clause describes <ontologyRepository> resource specific primitive behaviour for CRUD operations.

7.4.46.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

1) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

   a) The Hosting CSE shall also create the <semanticValidation> virtual child resource if the <ontologyRepository> resource is created successfully.

7.4.46.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.46.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.46.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

1) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

   a) The Hosting CSE shall also delete the <semanticValidation> virtual child resource if the <ontologyRepository> resource is deleted successfully.

7.4.47 Resource Type <ontology>

7.4.47.1 Introduction

The <ontology> resource is used to store the representation of a specific ontology. This representation may contain ontology descriptions in a variety of formats.

The detailed description can be found in clause 9.6.51 in oneM2M TS-0001 Functional Architecture [6].

Table 7.4.47.1-1: Data type definition of <ontology> resource

Data Type ID	File Name	Note
ontology	CDT-ontology-v3_11_0.xsd	

Table 7.4.47.1-2: Universal/Common Attributes of <ontology> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.47.1-3: Resource Specific Attributes of <ontology> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
description	O	O	xs:string	No default
ontologyFormat	M	O	m2m:semanticFormat	No default
ontologyContent	M	O	xs:anySimpleType	In case *ontologyFormat* is set as '1' (IRI), this attribute shall be of type xs:anyURI; otherwise, it shall be of type xs:base64Binary.
semanticOpExec	NP	O	m2m:sparql	No default

Table 7.4.47.1-4: Child Resources of <ontology> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.47.2 <ontology> resource specific procedures for CRUD operations

7.4.47.2.0 Introduction

This clause describes <ontology> resource specific primitive behaviour for CRUD operations.

7.4.47.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) The Hosting CSE shall check that the *ontologyContent* attribute conforms to the syntax defined by the *ontologyFormat* attribute.

   b) If the *ontologyContent* attribute does not conform, the Hosting CSE shall reject the request with a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

7.4.47.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.47.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

a) Primitive specific operation on Orig-1.0 "Compose Request primitive": The originator creates a request to update the *semanticOpExec* attribute. The value of this attribute is set to a SPARQL request that includes INSERT, DELETE, or DELETE/INSERT with conditional SPARQL statements as defined in the SPARQL query language [33].

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) If both *semanticOpExec* and *ontologyContent* attributes exist, the Receiver shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

   b) If the *semanticOpExec* attribute exists in the Request check that the syntax of its content corresponds to a valid SPARQL query request [33]. If the content does not correspond to a valid SPARQL query request, the Receiver shall reject the Request with a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

   c) If the *ontologyContent* attribute exists in the Request, check that the syntax of its content conforms to the syntax specified by the *ontologyFormat* attribute. If the content does not conform, the Receiver shall reject the Request with a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

2) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" in addition:

   a) If the *semanticOpExec* attribute exists in the Request, the Hosting CSE shall update the semantic triples in the *ontologyContent* attribute according to the SPARQL update request in the s*emanticOpExec* attribute. If the SPARQL update request cannot be executed, the Hosting CSE shall "create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "SPARQL_UPDATE_ERROR", otherwise proceed to step Recv-6.6.

7.4.47.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.48 Resource Type <semanticValidation>

7.4.48.1 Introduction

The <semanticValidation> virtual resource is used as the interface for semantic validation of an input <semanticDescriptor> resource.

The detailed description can be found in clause 9.6.52 in oneM2M TS-0001 Functional Architecture [6].

There are no common attributes, resource specific attributes or xsd file to <semanticValidation> resource because it is a virtual resource.

7.4.48.2 <semanticValidation> resource specific procedures for CRUD operations

7.4.48.2.0 Introduction

This clause describes <semanticValidation> resource specific primitive behaviour for CRUD operations.

7.4.48.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.48.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.48.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1 with the following exception:

1) Primitive specific operation on Orig-1.0 "Compose Request primitive": The resource representation in the ***Content*** parameter of the request message shall be set as the <semanticDescriptor> resource to be validated.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) Check if the ***Content*** parameter contains a valid resource representation conforming to the <semanticDescriptor> resource type. The Receiver shall perform the validity check as specified in clause 7.3.3.4.

   b) Check if the received <semanticDescriptor> resource contains the *ontologyRef* attribute with a valid URI value. If not, the Receiver shall generate a ***Response Status Code*** indicating a "CONTENTS_UNACCEPTABLE" error.

2) Skip Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" and Recv-6.6 "Announce/De-announce the resource". Instead, the Hosting CSE shall perform the following procedures according to oneM2M TS-0034 [50] to perform semantic validation:

   a) Access the semantic triples from the *descriptor* attribute of the received <semanticDescriptor> resource.

   b) Access the ontology referenced in the *ontologyRef* attribute of the received <semanticDescriptor> resource.

      - If the ontology referenced by the *ontologyRef* attribute is an external ontology, not locally hosted by the Hosting CSE, the Hosting CSE shall retrieve it using the corresponding protocol and identifier information specified in the *ontologyRef* attribute.

- If the referenced ontology cannot be retrieved within a reasonable time (as defined by a local policy), the Hosting CSE shall generate a ***Response Status Code*** indicating an "ONTOLOGY_NOT_AVAILABLE" error.

c) Retrieve any linked <semanticDescriptor> resources of the received <semanticDescriptor> resource according to the procedures defined in clause 7.4.34.2.2 with the following details:

- If the *relatedSemantics* attribute exists in the received <semanticDescriptor> resource, retrieve the linked <semanticDescriptor> resources following the URI(s) in the *relatedSemantics* attribute.
- If the *descriptor* attribute of the received <semanticDescriptor> resource contains triples with annotation property m2m:resourceDescriptorLink, retrieve the linked <semanticDescriptor> resources following the URI(s) in those triples.
- Repeat this step recursively to retrieve any further linked <semanticDescriptor> resources of the linked <semanticDescriptor> resources.
- If the linked <semanticDescriptor> resources cannot be retrieved within a reasonable time (as defined by a local policy), the Hosting CSE shall generate a Response Status Code indicating a "LINKED_SEMANTICS_NOT_AVAILABLE" error.

d) Retrieve the semantic triples from the *descriptor* attribute of the linked <semanticDescriptor> resource.

e) Retrieve the referenced ontologies of any linked <semanticDescriptor> resources following the URI(s) in *ontologyRef* attribute of the linked <semanticDescriptor> resources using the procedure defined in step 2e of clause 7.4.34.2.1. If the referenced ontologies cannot be retrieved within a reasonable time (as defined by a local policy), the Hosting CSE shall generate a ***Response Status Code*** indicating an "ONTOLOGY_NOT_AVAILABLE" error.

f) Combine all the semantic triples of the received and linked <semanticDescriptor> resources as the set of semantic triples to be validated, and combine all the referenced ontologies as the set of ontologies to validate the semantic triples against.

g) Check all the aspects of semantic validation according to clause 7.10.3 in oneM2M TS-0034 [50] based upon the combined semantic triples and referenced ontologies. If any problem occurs, the Hosting CSE shall generate a Response Status Code indicating a "INVALID_SEMANTICS" error and update the received <semanticDescriptor> resource with the *semanticValidated* attribute set to 'false'.

3) Primitive specific operation on Recv-6.7 "Create a success response":

a) The Hosting CSE shall update the received <semanticDescriptor> resource with the *semanticValidated* attribute set to 'true'.

4) As an optimization, the Hosting CSE may buffer the retrieved referenced ontologies and linked <semanticDescriptor> resources for the semantic process in future.

7.4.48.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.49 Resource Type <semanticMashupJobProfile>

7.4.49.1 Introduction

The <semanticMashupJobProfile> resource is used to store a Semantic Mashup Job Profile (SMJP).

The detailed description can be found in clause 6.3 in oneM2M TS-0034: Semantics Support [50] and clause 9.6.53 in oneM2M TS-0001: Functional Architecture [6].

Table 7.4.49.1-1: Data type definition of <semanticMashupJobProfile> resource

Data Type ID	File Name	Note
semanticMashupJobProfile	CDT-semanticMashupJobProfile-v3_11_0.xsd	

Table 7.4.49.1-2: Universal/Common Attributes of <semanticMashupJobProfile> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
accessControlPolicyIDs	O	O
expirationTime	O	O
dynamicAuthorizationConsultationIDs	O	O
announceTo	O	O
announcedAttribute	O	O
creator	O	NP

Table 7.4.49.1-3: Resource Specific Attributes of <semanticMashupJobProfile> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
memberFilter	M	O	m2m:sparql	No default.
smiID	O	O	m2m:listOfURIs	No default.
inputDescriptor	O	O	xs:base64Binary	The content shall be serialized using application/rdf+xml media type.
outputDescriptor	M	O	xs:base64Binary	The content shall be serialized using application/rdf+xml media type.
functionDescriptor	M	O	xs:base64Binary	The content shall be serialized using application/rdf+xml media type.

Table 7.4.49.1-4: Child Resources of <semanticMashupJobProfile> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<*subscription*>	[variable]	0..n	Clause 7.4.8
<*semanticMashupInstance*>	[variable]	0..n	Clause 7.4.50
<*semanticDescriptor*>	[variable]	0..1	Clause 7.4.34
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.49.2 <semanticMashupJobProfile> resource specific procedures for CRUD operations

7.4.49.2.0 Introduction

This clause describes <semanticMashupJobProfile> resource specific primitive behaviour for CRUD operations.

7.4.49.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exception:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) The Hosting CSE shall check that the *inputDescriptor, outputDescriptor* and *functionDescriptor* attributes conform to the RDF/XML syntax as defined in RDF 1.1 XML Syntax [34].

   b) The hosting CSE shall also check that the *memberFilter* attribute conforms to a valid SPARQL query request [33].

   c) If any of those attributes does not conform, the Hosting CSE shall generate a Response Status Code indicating a "NOT_ACCEPTABLE" error.

7.4.49.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.49.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) If any of those attributes (*inputDescriptor*, *outputDescriptor* and *functionDescriptor*) is being updated, The Hosting CSE shall check that the new values of those attributes being updated conform to the RDF/XML syntax as defined in RDF 1.1 XML Syntax [34].

   b) If the *memberFilter* attribute is being updated, the hosting CSE shall check that the new value of the *memberFilter* attribute conforms to a valid SPARQL query request [33].

   c) If any of the new values of those attributes does not conform, the Hosting CSE shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

7.4.49.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

   a) The Hosting CSE shall set a NULL value into the *smjpID* attribute of each <semanticMashupInstance> resource that is referenced by this resource's *smiID*.

7.4.50 Resource Type <semanticMashupInstance>

7.4.50.1 Introduction

The <semanticMashupInstance> resource is used to represent a Semantic Mashup Instance (SMI) that is instantiated based on a specific SMJP. The detailed description can be found in clause 6.4 in oneM2M TS-0034: Semantics Support [50] and clause 9.6.54 in oneM2M TS-0001: Functional Architecture [6].

Table 7.4.50.1-1: Data type definition of <semanticMashupInstance> resource

Data Type ID	File Name	Note
semanticMashupInstance	CDT-semanticMashupInstance-v3_11_0.xsd	

Table 7.4.50.1-2: Universal/Common Attributes of <semanticMashupInstance> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
accessControlPolicyIDs	O	O
expirationTime	O	O
dynamicAuthorizationConsultationIDs	O	O
announceTo	O	O
announcedAttribute	O	O
creator	O	NP

Table 7.4.50.1-3: Resource Specific Attributes of <semanticMashupInstance> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
smjpID	M	O	xs:anyURI	No default.
smjpInputParameter	M	O	xs:base64Binary	The content shall be serialized using application/rdf+xml media type.
memberStoreType	M	O	m2m:mashupMemberStoreType	No default.
mashupMember	O	O	m2m:mashupMembers	In case *memberStoreType* is set to URI_ONLY, this attribute shall only contain URIs. In case *memberStoreType* is URI_AND_VALUE, this attribute shall contain both URIs and their values. See clause 6.3.5.68.
resultGenType	M	O	m2m:mashupResultGenType	No default.
periodForResultGen	O	O	xs:duration	No default.

Table 7.4.50.1-4: Child Resources of <semanticMashupInstance> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<*semanticMashupResult*>	[variable]	0..n	Clause 7.4.51
<*semanticDescriptor*>	[variable]	0..1	Clause 7.4.34
<*subscription*>	[variable]	0..n	Clause 7.4.8
<*mashup*>	msp	0..1	Clause 7.4.52
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.50.2 <semanticMashupInstance> resource specific procedures for CRUD operations

7.4.50.2.0 Introduction

This clause describes <semanticMashupInstance> resource specific primitive behaviour for CRUD operations.

7.4.50.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

   a) The Hosting CSE shall check that the value of the *smjpID* attribute is not NULL and a valid SMJP can be retrieved by using the SMJP ID indicated in the *smjpID* attribute.

   b) The Hosting CSE shall check that the value of *smjpInputParameter* attribute conforms to the RDF/XML syntax as defined in RDF 1.1 XML Syntax [34] and that it meets the requirement described in the "*inputDescriptor*" attribute of the retrieved <*semanticMashupJobProfile*>.

   c) If any of the above checks fail, the Hosting CSE shall generate a ***Response Status Code*** indicating a "NOT_ACCEPTABLE" error.

d) The Hosting CSE shall use the SPARQL query specified in the *memberFilter* attribute of the retrieved <*semanticMashupJobProfile*> to discover mashup member resources for the <*semanticMashupInstance*> to be created. The hosting CSE shall conduct semantic resource discovery on one or more CSEs by using the SPARQL query statement as the semantics filter to identify qualified mashup members.

e) If not all of the required mashup members can be identified, the Hosting CSE shall generate a ***Response Status Code*** indicating a "MASHUP_MEMBER_NOT_FOUND" error.

f) If all the mashup members are identified, the *mashupMember* attribute shall be updated. If the *memberStoreType* attribute has the value of "URI_ONLY", then the URIs of those member resources identified (during step d) will be stored in the *mashupMember* attribute. If the *memberStoreType* attribute has the value of "URI_AND_VALUE", then the URIs as well as the values of those member resources identified (during step d) will be stored in the *mashupMember* attribute.

2) Primitive specific operation on Recv-6.5 “Create/Update/Retrieve/Delete/Notify operation is performed”:

a) If the *resultGenType* is set to WHEN_SMI_IS_CREATED the procedure for calculation of the semantic mashup result is performed and a new <*semanticMashupResult*> child resource is created.

b) If the *resultGenType* is set to PERIODICALLY, a timer is started with the given period, at which time the procedure for calculation of the semantic mashup result is performed and a new <*semanticMashupResult*> child resource is created.

7.4.50.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.50.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2 with the following exceptions:

1) Primitive specific operation on Recv-6.4 "Check validity of resource representation for the given resource type":

a) If the updated attribute in the Request message is *smjpInputParameter*, the Hosting CSE shall re-calculate the semantic mashup result. A <*semanticMashupInstance*> resource may have multiple <*semanticMashupResult*> child resources, which means that every time the Hosting CSE re-calculates the semantic mashup result, a new <*semanticMashupResult*> child resource shall be created to store the latest result.

b) If the updated attribute in the Request message is *memberStoreType*, the Receiver shall change the way mashup member resources are represented by the *mashupMember* attribute.

c) If the updated attribute in the Request message is *resultGenType*, the Receiver shall change the condition under which subsequent semantic mashup results are generated.

d) If the updated attribute in the Request message is *mashupMember* and *resultGenType* is "WHEN_A_MASHUP_MEMBER_IS_UPDATED", the Hosting CSE shall first update the attributes and then re-calculate the semantic mashup result.

7.4.50.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.51 Resource Type <mashup>

7.4.51.1 Introduction

<mashup> is a virtual resource and is the child resource of a <semanticMashupInstance> resource. When a RETRIEVE operation is sent to the <mashup> resource, it triggers a calculation and generation of the mashup result based on its parent resource <semanticMashupInstance>. The detailed description can be found in clause 6.5 in oneM2M TS-0034: Semantics Support [50] and clause 9.6.55 in oneM2M TS-0001: Functional Architecture [6].

7.4.51.2 <mashup> resource specific procedures for CRUD operations

7.4.51.2.0 Introduction

Only Retrieve operation shall be allowed on a <mashup> virtual resource. A Create, an Update, or a Delete operation on a <mashup> virtual resource shall not be supported.

7.4.51.2.1 Create

Originator:

The <mashup> resource shall not support Create operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.51.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-6.3 specified in clause 7.2.2.2 Generic Resource Request Procedure for Receiver, with the following primitive specific operations:

After Recv-6.3 "Check authorization of the Originator":

1) Check the *mashupMember* attribute of the parent <semanticMashupInstance> resource.

2) Collect latest data from each of the mashup members. If data cannot be collected from all of the mashup members, the Hosting CSE shall generate a ***Response Status Code*** indicating a "MASHUP_MEMBER_NOT_FOUND" error.

3) Retrieve the mashup function from the *functionDescriptor* attribute of the <semanticMashupJobProfile> resource that is specified by the *smjpID* attribute of the parent <semanticMashupInstance> resource.

4) Apply the mashup function with the data collected from the mashup members and produce a mashup result.

5) Create a <semanticMashupResult> child resource under the parent <semanticMashupInstance> resource, and use it to store the newly-generated mashup result.

6) Perform Recv-6.7 "Create a success response".

7) Perform Recv-6.8 and the procedure is terminated.

Note that, in case of the *resultGenType* attribute of the parent <semanticMashupInstance> resource is set to PERIODICALLY, steps 1) through 5) will be executed periodically by the Hosting CSE in order to generate the mashup result.

7.4.51.2.3 Update

Originator:

The <mashup> resource shall not support Update operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order:

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.51.2.4 Delete

Originator:

The <mashup> resource shall not support Delete operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.52 Resource Type <semanticMashupResult>

7.4.52.1 Introduction

<semanticMashupResult> resource stores the mashup result. It is a child resource of a <semanticMashupInstance> resource. A <semanticMashupResult> child resource shall be generated by a Hosting CSE when it executes a semantic mashup operation on the parent <semanticMashupInstance> resource. The further detailed description can be found in clause 6.6 in oneM2M TS-0034: Semantics Support [50] and clause 9.6.56 in oneM2M TS-0001: Functional Architecture [6].

Table 7.4.52.1-1: Data type definition of <semanticMashupResult> resource

Data Type ID	File Name	Note
semanticMashupInstance	CDT-semanticMashupResult-v3_11_0.xsd	

Table 7.4.52.1-2: Universal/Common Attributes of <semanticMashupResult> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	NP	NP
accessControlPolicyIDs	NP	NP
expirationTime	NP	NP
dynamicAuthorizationConsultationIDs	NP	NP
announceTo	NP	NP
announcedAttribute	NP	NP
creator	NP	NP

Table 7.4.52.1-3: Resource Specific Attributes of <semanticMashupResult> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
smjpInputParameter	NP	NP	xs:base64Binary	The content shall be serialized using application/rdf+xml media type.
mashupResultFormat	NP	NP	m2m:serializations	The permitted values include application/xml or application/json.
mashupResult	NP	NP	xs:base64Binary	No default.

Table 7.4.52.1-4: Child Resources of <semanticMashupResult> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<semanticDescriptor>	[variable]	0..1	Clause 7.4.34
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.52.2 <semanticMashupResult> resource specific procedures for CRUD operations

7.4.52.2.0 Introduction

This clause describes <semanticMashupResult> resource specific primitive behaviour for CRUD operations.

A <semanticMashupResult> resource shall be created by a Hosting CSE itself (the CSE that hosts the parent <semanticMashupInstance> resource and that executes the semantic mashup operation producing the mashup result) to store a newly-generated mashup result. Entities other than the Hosting CSE shall only be allowed to perform Retrieve and Delete operations on a <semanticMashupResult> resource.

7.4.52.2.1 Create

Originator:

The <semanticMashupResult> resource shall not support Create operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order.

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.52.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.52.2.3 Update

Originator:

The <semanticMashupResult> resource shall not support Update operations via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

If the request is received, the Receiver CSE shall execute the following steps in order.

1) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

2) "Send the Response primitive".

7.4.52.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.53 Resource Type <AEContactList>

7.4.53.1 Introduction

The <AEContactList> contains <AEContactListPerCSE> child resources, one for each CSE for which AE resource identifier information is being maintained. The <AEContactList> resource shall only be created as a child of <CSEBase> in the IN-CSE.

The detailed description can be found in clause 9.6.45 in oneM2M TS-0001 [6].

Table 7.4.53.1-1: Data type definition of <AEContactList> resource

Data Type ID	File Name	Note
AEContactList	CDT- AEContactList -v3_11_0.xsd	

Table 7.4.53.1-2: Universal/Common Attributes of <AEContactList> resource

Attribute Name
@resourceName
resourceType
resourceID
parentID
expirationTime
accessControlPolicyIDs
creationTime
lastModifiedTime
labels
dynamicAuthorizationConsultationIDs

Table 7.4.53.1-3: Resource Specific Attributes of <AEContactList> resource

Attribute Name	Data Type	Default Value and Constraints
numberImpactedCSEs	xs:nonNegativeInteger	No default

Table 7.4.53.1-4: Child Resources of <AEContactList> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<AEContactListPerCSE>	[variable]	0..n	Clause 7.4.54
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.53.2 <AEContactList> resource specific procedures for CRUD operations

7.4.53.2.0 Introduction

This clause describes <AEContactList> resource specific primitive behaviour for CRUD operations.

7.4.53.2.1 Create

Originator:

The <AEContactList> resource shall not be created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.53.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.53.2.3 Update

Originator:

The <AEContactList> resource shall not be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.53.2.4 Delete

Originator:

The <AEContactList> resource shall not be deleted via API.

Receiver:

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.54 Resource Type <AEContactListPerCSE>

7.4.54.1 Introduction

The <AEContactListPerCSE> resource contains AE resource identifier information which is being maintained. The <AEContactListPerCSE> is a child of a <AEContactList> resource in the IN-CSE and has no child resources.

The detailed description can be found in clause 9.6.46 in oneM2M TS-0001 [6].

Table 7.4.54.1-1: Data type definition of <AEContactListPerCSE> resource

Data Type ID	File Name	Note
AEContactListPerCSE	CDT- AEContactListPerCSE -v3_11_0.xsd	

Table 7.4.54.1-2: Universal/Common Attributes of <AEContactListPerCSE> resource

Attribute Name
@resourceName
resourceType
resourceID
parentID
expirationTime
accessControlPolicyIDs
creationTime
lastModifiedTime
labels
dynamicAuthorizationConsultationIDs

Table 7.4.54.1-3: Resource Specific Attributes of <AEContactListPerCSE> resource

Attribute Name	Data Type	Default Value and Constraints
CSE-ID	m2m:ID	No default
AE-IDList	m2m:listOfM2MID	No default

7.4.54.2 <AEContactListPerCSE> resource specific procedures for CRUD operations

7.4.54.2.0 Introduction

This clause describes <AEContactListPerCSE> resource specific primitive behaviour for CRUD operations.

7.4.54.2.1 Create

Originator:

The <AEContactListPerCSE> resource shall not be created via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.54.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.54.2.3 Update

Originator:

The <AEContactListPerCSE> resource shall not be updated via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.54.2.4 Delete

Originator:

The <AEContactListPerCSE> resource shall not be deleted via API.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1) If the request is received, the Receiver CSE shall execute the following steps in order:

   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

   b) "Send the Response primitive".

7.4.55 Resource Type <localMulticastGroup>

7.4.55.1 Introduction

The <localMulticastGroup> resource is used by a member-hosting CSE to indicate that this CSE is a member of a multicast group. The <localMulticastGroup> is created as a child resource of the <CSEBase> resource. There may be multiple <localMulticastGroup> resources under the same <CSEBase>.

The detailed description can be found in clause 9.6.44 in oneM2M TS-0001: Functional Architecture [6].

Table 7.4.55.1-1: Data type definition of <localMulticastGroup> resource

Data Type ID	File Name	Note
localMulticastGroup	CDT- localMulticastGroup-v3_11_0.xsd	

Table 7.4.55.1-2: Universal/Common Attributes of <localMulticastGroup> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
labels	O	O
accessControlPolicyIDs	O	O
expirationTime	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.55.1-3: Resource Specific Attributes of <localMulticastGroup> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
externalGroupID	O	O	m2m:externalID	No default
multicastAddress	M	O	m2m:ipAddress	No default
multicastGroupFanoutTarget	M	NP	xs:anyURI	The *multicastGroupFanout Target* should be uniquely assigned by the group Hosting CSE. It is set to /{groupHostingCSE-ID}/{fanout-segment}
memberList	M	O	m2m:listOfURIs	No default
responseTarget	O	O	xs:anyURI	No default
responseTimeWindow	O	O	xs:duration	No default
TMGI	O	O	m2m:TMGI	No default

7.4.55.2 <localMulticastGroup> resource specific procedures for CRUD operations

7.4.55.2.0 Introduction

This clause describes <localMulticastGroup> resource specific primitive behaviour for CRUD operations.

A <localMulticastGroup> resource shall be created by a Group Hosting CSE.

7.4.55.2.1 Create

Originator:

Primitive specific operation on Orig-1.0 "Compose Request primitive":

- The Group Hosting CSE shall set the *accessControlPolicyIDs* attribute as an <accessControlPolicy> resource with the Group Hosting CSE as the only entity that has CRUD privileges to the <localMulticastGroup>.

Primitive specific operation on Orig-6.0 "Process Response primitive":

- If at least two members responded successfully, a Multicast Group Information data object shall be created and maintained by the Group Hosting CSE. Details of Multicast Group Information are specified in clause 10.2.7.13 of oneM2M TS-0001 [6].

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation after Recv-1.0 "Check the syntax of received message" and before Recv-2.0 "Communication method?":

1) Comply with the multicast management protocol such as MLD or IGMP to join the multicast group corresponding to the *multicastAddress* in the request. If this procedure fails, the Receiver returns an error response with ***Response Status Code*** indicating "JOIN_MULTICAST_GROUP_FAILED".

No change for the remaining steps from the generic procedures in clause 7.2.2.1.

7.4.55.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.55.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.55.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic procedures in clause 7.2.2.2 except addition of the following to step Recv-6.5:

1) The receiver shall comply with the multicast management protocol such as MLD or IGMP to leave the multicast group. If this procedure fails, the Receiver returns an error response with ***Response Status Code*** indicating "LEAVE_MULTICAST_GROUP_FAILED".

7.4.56 Resource Type <multimediaSession>

7.4.56.1 Introduction

The <multimediaSession> resource represents a multimedia session between two AEs. The resource contains the session information which is used by the AEs to manage (e.g. establish, tear-down) the session using non-oneM2M protocols.

Table 7.4.56.1-1: Data type definition of <*multimediaSession*> resource

Data Type ID	File Name	Note
multimediaSession	CDT-multimediaSession-v3_11_0.xsd	

Table 7.4.56.1-2: Universal/Common Attributes of <*multimediaSession*> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announceTo	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.56.1-3: Resource Specific Attributes of <*multimediaSession*> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
sessionOriginatorID	M	NP	m2m:ID	
acceptedSessionDescriptions	NP	O	m2m:sessionDescriptions	
offeredSessionDescriptions	M	O	m2m:sessionDescriptions	
sessionState	NP	O	m2m:sessionState	This is either OFFLINE or ONLINE. The default value is OFFLINE.

Table 7.4.56.1-4: Child Resources of <*multimediaSession*> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.56.2 <multimediaSession> resource specific procedures for CRUD operations

7.4.56.2.0 Introduction

This clause describes <multimediaSession> resource specific primitive behaviour for CRUD operations.

7.4.56.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations.

The hosting CSE shall check if the targeted <*AE*> resource has the *sessionCapabilities* attribute present, and if not the Hosting CSE shall return the response primitive with a ***Response Status Code*** indicating "TARGET_HAS_NO_SESSION_CAPABILITY" error.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.56.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.56.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation on Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" with the following additional operations.

1) The Hosting CSE shall check if the UPDATE is modifying the *sessionState*, *offeredSessionDescriptions* or *acceptedSessionDescriptions* attributes.

   a) If yes, the Hosting CSE shall check if the *sessionState* attribute is set to ONLINE. If ONLINE, and the UPDATE is not modifying *sessionState* to OFFLINE but the UPDATE is modifying *offeredSessionDescriptions* or *acceptedSessionDescriptions* then the Hosting CSE shall reject the request and return the response primitive with a ***Response Status Code*** indicating "SESSION_IS_ONLINE" error. Otherwise the Hosting CSE shall perform the UPDATE.

No other changes from the generic procedures in clause 7.2.2.2.

7.4.56.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.57 Resource Type <triggerRequest>

7.4.57.1 Introduction

The <triggerRequest> resource is used to initiate a device trigger request. This resource type shall only be instantiated on an IN-CSE.

The successful creation of a <triggerRequest> resource results in the IN-CSE initiating a trigger request to a targeted device (e.g. 3GPP UE). The trigger will be routed to an application on the targeted device. The device is identified by M2M-Ext-ID and the application on the device is identified by Trigger-Recipient-ID. A pending trigger request can be replaced with a new trigger request by updating the <triggerRequest> resource. A pending trigger request can be recalled by deleting the <triggerRequest> resource.

Table 7.4.57.1-1: Data type definition of <triggerRequest> resource

Data Type ID	File Name	Note
triggerRequest	CDT-triggerRequest-v3_11_0.xsd	

Table 7.4.57.1-2: Universal/Common Attributes of <triggerRequest> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.57.1-3: Resource Specific Attributes of <triggerRequest> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
M2M-Ext-ID	M	NP	m2m:externalID	No default
Trigger-Recipient-ID	M	O	m2m:triggerRecipientID	No default
triggerPurpose	O	O	m2m:triggerPurpose	establishConnection
triggerStatus	NP	NP	m2m:triggerStatus	No default
triggerValidityTime	M	O	xs:duration	No default
triggerInfoAE-ID	O	O	m2m:ID	No default
triggerInfoAddress	O	O	xs:anyURI	No default
triggerInfoOperation	O	O	m2m:operation	No default
targetedResourceType	O	O	m2m:resourceType	No default
triggerReference	NP	NP	xs:string	No default This is generated by the Hosting CSE

Table 7.4.57.1-4: Child Resources of <triggerRequest> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8
<transaction>	[variable]	0..n	Clause 7.4.61

7.4.57.2 <triggerRequest> resource specific procedures for CRUD operations

7.4.57.2.0 Introduction

This clause describes <triggerRequest> resource specific primitive behaviour for CRUD operations.

7.4.57.2.1 Create

Originator:

The Originator shall use the steps Orig-1.0, Orig-2.0, and Orig-3.0 as described in clause 7.2.2.1.

Receiver:

The Receiver shall use the steps Recv-1.0 to Recv-10.0 as described in clause 7.2.2.2.

While processing the <triggerRequest> Create primitive, the Receiver shall detect the following types of errors and send a corresponding status code to the Originator.

- If the Originator specifies a *Trigger-Recipient-ID* value in the Create primitive for a Registree AE or CSE, and the *triggerEnable* attribute of the Registree's <AE> or <remoteCSE> resource has a value of false, the Receiver shall generate a ***Response Status Code*** indicating "TRIGGERING_DISABLED_FOR_RECIPIENT".

While processing the <triggerRequest> Create primitive the Receiver shall determine which NSE to forward the trigger request to based on locally provisioned information or based on a DNS lookup of the M2M-Ext-ID attribute of the <triggerRequest>. If an NSE cannot be determined, the Receiver shall set the *triggerStatus* attribute to ERROR_NSE_NOT_FOUND. Otherwise, the Receiver shall continue to process the trigger request and set the *triggerStatus* attribute to PROCESSING.

To continue processing the request, the Receiver shall submit a trigger request to the NSE via the Mcn triggering procedure as defined in clause 9. The message shall contain information needed by the NSE to generate a trigger request for the corresponding underlying network. For a 3GPP trigger request, the required information within the trigger request message is captured in clause 7.5.1 of oneM2M TS-0026 [43].

Upon receipt of trigger response(s) from the NSE, the Receiver shall set the *triggerStatus* attribute of the <triggerRequest> resource:

- If the Receiver receives a confirmation from the NSE that the trigger was accepted, the Receiver shall set the *triggerStatus* attribute to TRIGGER_SUBMITTED.
- If the Receiver receives an indication that the trigger request was successfully delivered, the Receiver shall set the *triggerStatus* attribute to TRIGGER_DELIVERED.
- If the Receiver receives an indication that the trigger request was not accepted or the delivery was not successful, the Receiver shall set the *triggerStatus* attribute to TRIGGER_FAILED.
- If the Receiver receives an indication that the trigger request expired before completion the Receiver shall set the *triggerStatus* attribute to TRIGGER_EXPIRED.

7.4.57.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.57.2.3 Update

The following procedure replaces an outstanding trigger request that is still being processed by an underlying network with an updated trigger request.

Originator:

The Originator shall use the steps Orig-1.0, Orig-2.0, and Orig-3.0 as described in clause 7.2.2.1.

Receiver:

The Receiver shall use the steps Recv-1.0 to Recv-10.0 as described in clause 7.2.2.2.

The Originator shall provide the <triggerRequest> resource representation to the Receiver IN-CSE. While processing the <triggerRequest> Update primitive, the Receiver shall detect the following types of errors and send a corresponding status code to the Originator.

- If the value of *triggerStatus* of the existing <triggerRequest> is PROCESSING, the Receiver shall continue to process the Update request. Otherwise, the Receiver shall generate a ***Response Status Code*** indicating "UNABLE_TO_REPLACE_REQUEST".
- If the Originator specifies a *Trigger-Recipient-ID* value in the Update primitive for a Registree AE or CSE, and the *triggerEnable* attribute of the Registree's <AE> or <remoteCSE> resource has a value of false, the Receiver shall generate a ***Response Status Code*** indicating "TRIGGERING_DISABLED_FOR_RECIPIENT".

While processing the <triggerRequest> Update primitive, the Receiver shall forward the trigger replace request to the same NSE that the trigger request was forwarded to when the <triggerRequest> was created. If the NSE cannot be reached, the Receiver shall set the *triggerStatus* attribute to ERROR_NSE_NOT_FOUND.

To continue processing the request, the Receiver shall submit the trigger request to the NSE via the Mcn triggering procedure defined in clause 9. The message shall contain information needed by the NSE to replace the trigger request for the corresponding underlying network. For a 3GPP trigger replace request, the required information within the trigger request message is captured in clause 7.5.2 of oneM2M TS-0026 [43].

Upon receipt of a successful trigger replace response from the NSE, the Receiver shall generate a ***Response Status Code*** indicating "UPDATED". Otherwise, the Receiver shall generate a ***Response Status Code*** indicating "UNABLE_TO_REPLACE_REQUEST".

7.4.57.2.4 Delete

Originator:

The Originator shall issue a request to the Receiver IN-CSE to delete the <triggerRequest> resource. The Originator shall use the steps Orig-1.0, Orig-2.0, and Orig-3.0 as described in clause 7.2.2.1.

Receiver:

The Receiver shall use the steps Recv-1.0 to Recv-10.0 as described in clause 7.2.2.2.

While processing the <triggerRequest> Delete primitive, the Receiver shall detect the following types of errors and send a corresponding status code to the Originator:

- If the value of *triggerStatus* of the existing <triggerRequest> is PROCESSING, the Receiver shall continue to process the Delete request. Otherwise, the Receiver shall generate a ***Response Status Code*** indicating "UNABLE_TO_RECALL_REQUEST".

While processing the <triggerRequest> Delete primitive, the Receiver shall send a request to the same NSE that the trigger request was sent to when the <triggerRequest> was created. If the NSE cannot be reached, the Receiver shall set the *triggerStatus* attribute to ERROR_NSE_NOT_FOUND.

To continue processing the request, the Receiver shall submit the trigger recall request to the NSE via the Mcn triggering procedure defined in clause 9. The message shall contain information needed by the NSE to recall the trigger request for the corresponding underlying network. For a 3GPP trigger recall request, the required information within the trigger recall request message is captured in clause 7.5.2 of oneM2M TS-0026 [43].

Upon receipt of a successful trigger recall response from the NSE, the Receiver shall delete the <triggerRequest> resource and generate a ***Response Status Code*** indicating "DELETED". Otherwise, the Receiver shall not delete the <triggerRequest> resource and instead generate a ***Response Status Code*** indicating "UNABLE_TO_RECALL_REQUEST".

7.4.58 Resource Type <crossResourceSubscription>

7.4.58.1 Introduction

The <crossResourceSubscription> resource is used to support cross-resource subscription and cross-resource notification. A cross-resource subscription is made on multiple target resources. A cross-resource notification shall be generated if and only if expected events on all target resources occur within a designated time window. When notifications from all target resources occur within a specified time window the Hosting CSE shall issue a cross-resource notification.

The detailed description can be found in clause 9.6.58 in oneM2M TS-0001 [6].

Table 7.4.58.1-1: Data type definition of <crossResourceSubscription> resource

Data Type ID	File Name	Note
crossResourceSubscription	CDT-crossResourceSubscription-v3_11_0.xsd	

Table 7.4.58.1-2: Universal/Common Attributes of <crossResourceSubscription> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	M	NP

Table 7.4.58.1-3: Resource Specific Attributes of <crossResourceSubscription> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
regularResourcesAsTarget	O	O	m2m:listOfURIs	No default
subscriptionResourcesAsTarget	O	O	m2m:listOfURIs	No default
timeWindowType	M	O	m2m:timeWindowType	PERIODICWINDOW
timeWindowSize	M	O	m2m:absRelTimestamp	No default, but shall be a relative timestamp (i.e. a time duration)
eventNotificationCriteriaSet	O	O	m2m:eventNotificationCriteriaSet	Default behaviour is notification on Update_of_Resource
notificationEventCat	O	O	m2m:eventCat	No default
expirationCounter	O	O	xs:positiveInteger	No default
notificationURI	M	O	list of xs:anyURI	No default
subscriberURI	O	NP	m2m:ID	No default

Table 7.4.58.1-4: Child Resources of <crossResourceSubscription> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<notificationTargetMgmtPolicyRef>	[variable]	0..n	Clause 7.4.30
<notificationTargetSelfReference>	ntsr	1	Clause 7.4.33
<transaction>	[variable]	0..n	Clause 7.4.61
<schedule>	notificationSchedule	0..1	Clause 7.4.9

7.4.58.2 <crossResourceSubscription> resource specific procedures for CRUD operations

7.4.58.2.0 Introduction

This clause describes <crossResourceSubscription> resource specific primitive behaviour for CRUD operations.

7.4.58.2.1 Create

Originator:

The following are changes to the Originator procedures described in clause 7.2.2.1:

1) Orig-1.0 When composing a request primitive, the Originator shall include *regularResourcesAsTarget* and/or *subscriptionResourcesAsTarget* attributes in the resource representation of the <crossResourceSubscription> in the content of the primitive. If *regularResourcesAsTarget* attribute is included, *eventNotificationCriteriaSet* attribute shall be included. If *eventNotificationCriteriaSet* contains only one *eventNotificationCriteria*, this *eventNotificationCriteria* shall be applied to all regular resources included in *regularResourcesAsTarget* attribute; otherwise, *eventNotificationCriteriaSet* shall contain the same number of *eventNotificationCriteria* elements as the number of regular target resources contained in *regularResourcesAsTarget* and each eventNotificationCriteria element shall be sequentially applied to corresponding target resource as listed in the *regularResourcesAsTarget*.

Receiver:

The following are changes to the receiver procedures described in clause 7.2.2.2:

1) Recv-6.5: The following steps are in addition to the generic Create procedures defined in clause 7.3.3.5:

   a) The request shall be rejected with a "BAD_REQUEST" ***Response Status Code*** if at least one of *regularResourcesAsTarget* or *subscriptionResourcesAsTarget* attributes is not present in the request.

   b) If *regularResourcesAsTarget* is included, the Hosting CSE shall send a CREATE <subscription> request message to each target resource indicated by *regularResourcesAsTarget*.

i) In the new CREATE <subscription> request, the receiver shall use the ***From*** of the current CREATE request. For this <subscription> to be created:

1) *eventNotificationCriteria* attribute shall use the corresponding entry included in *eventNotificationCriteriaSet* attribute of this <crossResourceSubscription> being created.

2) *notificationURI* attribute shall be set to the resource identifier of this <crossResourceSubscription> resource being created.

3) *associatedCrossResourceSub* attribute shall be set to the resource identifier of this <crossResourceSubscription> resource being created.

4) *notificationEventCat* attribute shall be set to the same value in the <crossResourceSubscription> resource representation.

ii) If any <subscription> for a target resource cannot be successfully created, the receiver shall send an unsuccessful response with a "CROSS_RESOURCE_OPERATION_FAILURE" ***Response Status Code*** to the Originator; the receiver shall also delete already created <subscription> resources at other target resources that were created based on the presence of *regularResourcesAsTarget*.

c) If *subscriptionResourcesAsTarget* is included, the Hosting CSE shall add the resource identifier of this <crossResourceSubscription> resource to the *associatedCrossResourceSub* attribute of each <subscription> resource indicated in *subscriptionResourcesAsTarget* by issuing an UPDATE request to the <subscription> resource host.

iii) In the UPDATE request, the receiver shall use the ***From*** *parameter* from the current CREATE request.

iv) *notificationURI* attribute shall be updated to include the resource identifier of this <crossResourceSubscription> resource being created.

v) If any <subscription> for a target resource cannot be successfully updated, the receiver shall send an unsuccessful response with a "CROSS_RESOURCE_OPERATION_FAILURE" ***Response Status Code*** to the Originator; the Hosting CSE shall also remove itself from any already successfully associated <subscription> resources using the procedures in clause 7.4.8.2.4 and also delete any already-created <subscription> resources at other target resources.

d) Once the <crossResourceSubscription> resource is created, the Hosting CSE shall start the time window if the *timeWindowType*=PERIODICWINDOW; if *timeWindowType*=SLIDINGWINDOW, the Hosting CSE shall start the time window after the first notification is received from a Target Resource Hosting CSE.

7.4.58.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.58.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following are changes to the receiver procedures described in clause 7.2.2.2.

1) Recv-6.5: The following steps are in addition to the generic Update procedures defined in clause 7.3.3.7:

a) If *regularResourcesAsTarget* is updated, the Hosting CSE shall perform the following tasks:

i) If a target resource has been removed in the new *regularResourcesAsTarget* attribute value delete the associated <subscription> child resource using the procedure in clause 7.4.8.2.4.

ii) If the updated *regularResourcesAsTarget* attribute value contains new target resources, the Hosting CSE shall send a Create <subscription> request message to each new target resource as described in clause 7.4.58.2.1.

b) If *subscriptionResourcesAsTarget* is updated, the Hosting CSE shall perform the following tasks:

iii) If a <subscription> resource has been removed in the new *subscriptionResourcesAsTarget* attribute value the Hosting CSE shall update the <subscription> resource using the procedure in clause 7.4.8.2.4 to remove this <crossResourceSubscription> from the <subscription> resource's *associatedCrossResourceSub* attribute.

iv) If the updated *subscriptionResourcesAsTarget* attribute value contains new a <subscription>resource, the Hosting CSE shall add the resource identifier of this <crossResourceSubscription> resource to the *associatedCrossResourceSub* attribute of each <subscription> resource indicated in *subscriptionResourcesAsTarget* as described in clause 7.4.58.2.1.

c) If *eventNotificationCriteriaSet* is updated, the Hosting CSE shall update the *eventNotificationCriteria* of each previously created <subscription> child resource of the targets listed in the *regularResourcesAsTarget* attribute to reflect the received *eventNotificationCriteria* content using the procedures in clause 7.4.8.2.3.

d) If *timeWindowSize* or *timeWindowType* is updated in the resource representation the receiver shall restart the timer as described in clause 7.4.58.2.1.

e) If any of the Update procedures are unsuccessful the receiver shall send an unsuccessful response with a "CROSS_RESOURCE_OPERATION_FAILURE" ***Response Status Code*** to the Originator; the receiver shall also restore all resources to the states they were in prior to this request.

7.4.58.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

The following are changes to the receiver procedures described in clause 7.2.2.2:

1) Recv-6.5: The following steps are in addition to the generic Delete procedures defined in clause 7.3.3.5:

a) The Hosting CSE shall delete the previously created <subscription> child resource of each target resource indicated in the *regularResourcesAsTarget* attribute. The Receiver shall use the ***From*** of the current request for these requests.

b) The Hosting CSE shall UPDATE the <subscription> resource of each target resource indicated in the *subscriptionResourcesAsTarget* attribute using the procedure in clause 7.4.8.2.3 to remove the resource identifier of this <crossResourceSubscription> from the <subscription> resource's *associatedCrossResourceSub* attribute. The Receiver shall use the ***From*** of the current request for these requests.

7.4.59 Resource Type <backgroundDataTransfer>

7.4.59.1 Introduction

The <backgroundDataTransfer> resource is used to request that the IN-CSE negotiates a background data transfer for a set of field nodes, with the Underlying Network. The resource attributes provide the characteristics of the background data transfer, optional guidance for transfer policy selection and the field nodes involved with the data transfer. Additional description of the <backgroundDataTransfer> resource is contained in clauses 9.6.60 and 10.2.20 of oneM2M TS-0001 [6]. The corresponding procedures over the Mcn reference point are described in oneM2M TS-0026 [43].

Table 7.4.59.1-1: Data type definition of <backgroundDataTransfer> resource

Data Type ID	File Name	Note
backgroundDataTransfer	CDT-backgroundDataTransfer-v_3_7_0.xsd	

Table 7.4.59.1-2: Universal/Common Attributes of <backgroundDataTransfer> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
announcedAttribute	O	O
dynamicAuthorizationConsultationIDs	O	O
creator	O	NP

Table 7.4.59.1-3: Resource Specific Attributes of <backgroundDataTransfer> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
volumePerNode	M	O	xs:positiveInteger	No default
numberOfNodes	M	O	xs:positiveInteger	No default
desiredTimeWindow	M	O	m2m:scheduleEntry	No default. The actual value may be modified by the CSE
transferSelectionGuidance	O	O	m2m:transferSelectionGuidance	No default. If not provided the CSE will set this value based on local policy
geographicInformation	O	O	m2m:locationRegion	No default
groupLink	O	O	xs:anyURI	No default
memberIDs	O	O	list of xs:anyURI	No default

Table 7.4.59.1-4: Child Resources of <backgroundDataTransfer> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8

7.4.59.2 <backgroundDataTransfer> resource specific procedures for CRUD operations

7.4.59.2.0 Introduction

This clause describes <backgroundDataTransfer> resource specific primitive behaviour for CRUD operations.

7.4.59.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.59.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.59.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.59.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.60 Resource Type <transactionMgmt>

7.4.60.1 Introduction

This resource is used to initiate and manage the atomic and consistent processing of a transaction consisting of multiple oneM2M request primitives. The detailed description can be found in clause 9.6.47 of oneM2M TS-0001 [6].

Table 7.4.60.1-1: Data type definition of <transactionMgmt> resource

Data Type ID	File Name	Note
transactionMgmt	CDT-transactionMgmt-v3_11_0.xsd	

Table 7.4.60.1-2: Universal/Common Attributes of <transactionMgmt> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
creator	O	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.60.1-3: Resource Specific Attributes of <transactionMgmt> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
transactionLockTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy.
transactionExecuteTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy.
transactionCommitTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy.
transactionExpirationTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy.
transactionMode	O	NP	m2m:transactionMode	CSE_CONTROLLED
transactionLockType	O	NP	m2m:transactionLockType	BLOCK_ALL
transactionControl	NP	O	m2m:transactionControl	INITIAL
transactionState	NP	NP	m2m:transactionState	This value is set by the Hosting CSE to indicate the current state of the transaction.
transactionMaxRetries	O	O	xs:nonNegativeInteger	0 (No Retries)
transactionMgmtHandling	O	O	m2m:transactionMgmtHandling	DELETE
requestPrimitives	M	NP	m2m:aggregatedRequest	No default
responsePrimitives	NP	NP	m2m:aggregatedResponse	This value is set by the Hosting CSE.

Table 7.4.60.1-4: Child Resources of <transactionMgmt> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8

7.4.60.2 <transactionMgmt> resource specific procedures for CRUD operations

7.4.60.2.0 Introduction

This clause describes <transactionMgmt> resource specific primitive behaviour for CRUD operations.

7.4.60.2.1 Create

Originator:

The following are changes to the Originator procedures described in clause 7.2.2.1.

Orig-1.0 When composing a request primitive, the Originator shall include the *requestPrimitives* attribute in the resource representation of the <transactionMgmt> in the content of the primitive. Each request primitive in the *requestPrimitives* attribute shall be created using the procedures described in clause 7.2.2.1.

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions.

1) Recv-6.4:

   a) The receiver shall set the *transactionControl* value to INITIAL.

   b) If any of the ***From*** parameters contained in the *requestPrimitives* attribute of the received <transactionMgmt> resource is not equal to the Originator of the received request primitive, the receiver shall generate a ***Response Status Code*** indicating "BAD_REQUEST".

NOTE: Process the <transactionMgmt> resource as described in clause 10.2.18.1 of oneM2M TS-0001 [6] after Recv-6.7.

7.4.60.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.60.2.3 Update

Originator:

The following are changes to the Originator procedures described in clause 7.2.2.1:

1) Orig-1.0 When composing a request primitive, if the originator changes the *transactionControl* value the originator shall use allowed values as specified in Table 10.2.18.1-1 of oneM2M TS-0001 [6].

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions:

1) Recv-6.3:

   - If there is a *transactionControl* value in the update primitive and if the Originator does not match the *creator* of the <transactionMgmt> resource the receiver shall generate a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE".

2) Recv-6.4:

- If there is a *transactionControl* value in the update primitive:

  i) If *transactionMode* has the value "CSE_CONTROLLED" then the Receiver shall generate a ***Response Status Code*** indicating "BAD_REQUEST".

  ii) If the *transactionState* value indicates that the previous transition [as shown by the *transactionControl* value in the <transactionMgmt> resource] is not complete the Receiver shall generate a ***Response Status Code*** indicating "TRANSACTION_PROCESSING_IS_INCOMPLETE".

  iii) If the *transactionControl* value in the request primitive does not transition to values specified in Table 10.2.18.1-1 of oneM2M TS-0001 [6] the Receiver shall generate a ***Response Status Code*** indicating "ILLEGAL_TRANSACTION_STATE_TRANSITION_ATTEMPTED".

NOTE: Process the <transactionMgmt> resource as described in clause 10.2.18.1 of oneM2M TS-0001 [6] after Recv-6.7.

7.4.60.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions.

1) Recv-6.5:

- The receiver shall check that the *transactionState* is either COMMITTED or ABORTED before deleting the <transactionMgmt> resource. To commit or abort the transaction the Receiver shall follow the procedure defined in clause 10.2.18.1 of oneM2M TS-0001 [6].

7.4.61 Resource Type <transaction>

7.4.61.1 Introduction

This resource is used to initiate and manage the atomic and consistent processing of a single oneM2M request primitive of a oneM2M transaction.

The detailed description can be found in clause 9.6.48 in oneM2M TS-0001 [6].

A <transaction> create request may be originated by a CSE that hosts a <transactionMgmt> resource. Alternatively, a <transaction> resource may be used independent of a <transactionMgmt> resource when an AE creates individual <transaction> resources itself.

Table 7.4.61.1-1: Data type definition of <transaction> resource

Data Type ID	File Name	Note
transaction	CDT-transaction-v3_11_0.xsd	

Table 7.4.61.1-2: Universal/Common Attributes of <transaction> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	O	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
creator	O	NP
dynamicAuthorizationConsultationIDs	O	O

Table 7.4.61.1-3: Resource Specific Attributes of <transaction> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
transactionID	M	NP	xs:string	No default
transactionControl	NP	O	m2m:transactionControl	LOCK
transactionState	NP	NP	m2m:transactionState	This value is set by the Hosting CSE to indicate the current state of the transaction
transactionLockTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy
transactionExecuteTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy
transactionCommitTime	O	NP	m2m:timestamp	No default. This value can be set or modified by the Hosting CSE according to local policy
transactionLockType	O	NP	m2m:transactionLockType	BLOCK_ALL
m2m:requestPrimitive	M	NP	Anonymous data type defined in the requestPrimitive declaration	No default. The m2m:requestPrimitive element is defined in CDT-requestPrimitive-v3_11_0.xsd See note
m2m:responsePrimitive	NP	NP	Anonymous data type defined in the responsePrimitive declaration	This value is set by the Hosting CSE. The m2m:responsePrimitive element is defined in CDT-responsePrimitive-v3_11_0.xsd See note
NOTE: The element name shall contain the namespace prefix.				

Table 7.4.61.1-4: Child Resources of <transaction> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.4.8

7.4.61.2 <transaction> resource specific procedure on CRUD operations

7.4.61.2.0 Introduction

This clause describes <transaction> resource specific primitive behaviour for CRUD operations.

7.4.61.2.1 Create

Originator:

No change from the generic procedures described in clause 7.2.2.1.

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions.

1) Recv-6.4: The following steps are in addition to the generic Create procedures defined in clause 7.3.3.5:

   - The receiver shall set the *transactionControl* value to LOCK.
   - If the ***From*** parameter contained in the *m2m:requestPrimitive* attribute of the received <transaction> resource is not equal to the Originator of the received request primitive, the receiver shall generate a ***Response Status Code*** indicating "BAD_REQUEST".

2) Recv-6.5: The following steps are in addition to the generic Create procedures defined in clause 7.3.3.5:

   - Process the <transaction> resource as described in clause 10.2.18.1 of oneM2M TS-0001 [6].

7.4.61.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

7.4.61.2.3 Update

Originator:

The following are changes to the Originator procedures described in clause 7.2.2.1.

- Orig-1.0 When composing a request primitive, if the originator changes the *transactionControl* value the originator shall use allowed values as specified in Table 10.2.18.1-1 of oneM2M TS-0001 [6].

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions.

1) Recv-6.3:

   - If there is a *transactionControl* value in the update primitive and if the Originator does not match the *creator* of the <transaction> resource the receiver shall generate a ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE".

2) Recv-6.4:

   - If the *transactionControl* value is in the update primitive:
     - If the *transactionControl* value does not transition to values specified in Table 10.2.18.1-1 of oneM2M TS-0001 [6] the Receiver shall generate a ***Response Status Code*** indicating "ILLEGAL_TRANSACTION_STATE_TRANSITION_ATTEMPTED".

- If *transactionState* is not equal to the value of *transactionControl* being replaced by this Update operation the Receiver shall generate a ***Response Status Code*** indicating "TRANSACTION_PROCESSING_IS_INCOMPLETE".

3) Recv-6.5: The following steps are in addition to the generic Update procedures defined in clause 7.3.3.7:

   - Process the <transaction> resource as described in clause 10.2.18.1 of oneM2M TS-0001 [6].

7.4.61.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Same as the generic operations detailed in clause 7.2.2.2 with the following additions.

1) Recv-6.5:

   - The Receiver shall check that the *transactionState* is either COMMITTED or ABORTED before deleting the <transaction> resource. To commit or abort the transaction the Receiver shall follow the procedure defined in clause 10.2.18.1 of oneM2M TS-0001 [6].

7.5 Primitive-specific procedures and definitions

7.5.1 Notification data object and procedures

7.5.1.1 Notification data object

Notification procedures represent a special case of the generic procedures defined in clause 7.2.2, where the ***Operation*** parameter of the request primitive is set to value "N" (Notify). In this case, the request primitive is referred to as *Notify request primitive*, and the associated response primitive is denoted as *Notify response primitive*.

A Notify request primitive shall convey a special notification data object in its ***Content*** parameter. This notification data object has no resource type representation in the oneM2M TS-0001 [6], since it does not represent a resource accessible by any M2M entities. The data type of the notification data object is defined in the tables below. The first column of Table 7.5.1.1-2 defines the permitted names the root element the notification data object can take with the data type listed in the third column.

Table 7.5.1.1-1: Data Type Definition of notification data object

Data Type ID	File Name	Note
notification	CDT-notification-v3_11_0.xsd	

Table 7.5.1.1-2: Data Types for notification data objects

Root Element Name	Request Optionality	Data Type	Default Value and Constraints
	N		
m2m:notification	O	m2m:notification	
m2m:aggregatedNotification	O	m2m:aggregatedNotification	
m2m:responsePrimitive	O	Anonymous data type defined in the responsePrimitive declaration	
m2m:securityInfo	O	m2m:securityInfo	

When an Originator sends a Notify primitive to an AE Receiver, it shall use one of the serializations specified in that <AE>'s *contentSerialization* attribute. When an Originator sends a Notify primitive to a CSE, as the receiver or a transit

CSE, it shall use one of the serializations specified in that <remoteCSE>'s *contentSerialization* attribute. If there is no *contentSerialization* value specified the Originator may use any serialization format.

7.5.1.2 Notification procedures

7.5.1.2.1 Introduction

Notification procedures shall be employed for the following use cases:

- to notify Receiver(s) of modifications of a resource for an associated <subscription> resource;
- to request Receiver(s) to perform resource subscription verification;
- to notify deletion of the <subscription> resource;
- to notify Receiver(s) for Asynchronous Non-blocking Request;
- to notify Receiver(s) of modifications of a resource when the subscription relationship is established through the <group> resource;
- to send the response corresponding to a request delivered via service layer long polling (clause 7.4.22.2.2 Retrieve <pollingChannelURI>);
- to notify Receiver(s)(i.e. IPE) for on-demand discovery request;
- to notify Receiver(s) of the missing Time Series Data points for an associated <subscription> resource;
- to notify Receiver(s) of a security related request (e.g. dynamic authorization and end-to-end security);
- to notify Receivers that an AE has changed registration point;
- to notify an IN-CSE that the Originator has a new/updated/deleted reference to an Application Entity Resource identifier;
- to notify Receiver(s) of a cross-resource notification generated by a <crossResourceSubscription> Hosting CSE;
- to notify a Receiver that a triggered update on the subscribed-to resource has been blocked and retargeted to the Receiver.

The following clauses specify the notification procedures for each of the above use cases.

7.5.1.2.2 Notification for <subscription> resources

When the notification message is forwarded or aggregated by transit CSEs, the Originator or a transit CSE shall check whether there are notification policies to enforce between subscription resource Hosting CSE and the notification target. In that case, the transit CSE as well as the Originator shall process Notify request primitive(s) by using the corresponding policy and send processed Notify request primitive(s) to the next CSE with notification policies related to the enforcement so that the transit CSE is able to enforce the policy defined by the subscriber. The notification policies related to the enforcement at this time is verified by using the subscription reference in the Notify request primitive. In the notification policies, the *latestNotify* attribute is only enforced in the transit CSE as well as the Originator.

If ***Event Category*** parameter is set to 'latest' in the notification request primitive, the transit CSE as well as Originator shall cache the most recent Notify request. That is, if a new Notify request is received by the CSE with a subscription reference that has already been buffered for a pending Notify request, the newer Notify request will replace the buffered older Notify request.

Originator:

When an event is generated, the Originator shall execute the following steps in order:

Step 1.0 Check the *eventNotificationCriteria* attribute of the <subscription> resource associated with the modified resource:

- If the *eventNotificationCriteria* attribute is set, then the Originator shall check whether the corresponding event matches with the event criteria. If multiple matching conditions of different types (i.e. different condition tags) are present in the *eventNotificationCriteria* attribute, then the combined condition shall be derived by applying the logical operation specified by the *filterOperation* condition. By default the logical AND operation shall be used if the *filterOperation* condition is not present.

- If *notificationEventType* is not set within the *eventNotificationCriteria* attribute and the *operationMonitor* is also not present, the Originator shall use the default setting of 'Update_of_Resource' to compare against the event.

- If the *notificationEventType* has the value 'Create_of_Direct_Child_Resource' and the *childResourceType* condition is also present, then the matching event shall only be detected if one of the child resource types present in the list has been created. If the *childResourceType* condition is not present then a matching event is generated whenever any child resource is created.

- If the *notificationEventType* has either an explicit or default value of 'Update_of_Resource' and the *attribute* condition is also present then the matching event shall only be detected if one of the attributes in the list has been updated. If the *attribute* condition is not present then a matching event is generated whenever any attribute has been updated.

- If the event matches, go to the step 2.0. Otherwise, the Originator shall discard the corresponding event.

- If the *eventNotificationCriteria* attribute is not configured, the Originator shall use the default setting of 'Update_of_Resource' for the *notificationEventType* and then continue with the step 2.0.

Step 2.0 The Originator shall check the notification policy as described in the below steps, but the notification policy may be checked in different order. After checking the notification policy in step 2.0 (i.e. from step 2.1 to step 2.6), then continue with step 3.0.

Step 2.1 The Originator shall determine the type of the notification per the *notificationContentType* attribute. The possible values of for *notificationContentType* attribute are 'Modified Attributes', 'All Attributes', 'ResourceID' or 'Trigger Payload'. This attribute may be used jointly with *eventType* attribute in the *eventNotificationCriteria* to determine if it is the attributes/resourceID of the subscribed-to resource or the attributes/resourceID of the child resource of the subscribed-to resource that shall be returned in the notification:

- If the value of *notificationContentType* is set to 'Modified Attributes', the Notify request primitive shall include the partial resource containing modified attribute(s) only (Refer to clause 7.2.1.2 for response content description).

- If the value of *notificationContentType* is set to 'All Attributes', the Notify request primitive shall include the complete resource with all attributes (Refer to clause 7.2.1.2 for response content description).

- If the value of *notificationContentType* is set to 'ResourceID', the Notify request primitive shall include the URI of the resource (Refer to clause 7.2.1.2 for response content description).

- If the value of *notificationContentType* is set to 'Trigger Payload', the Notify request primitive shall include the trigger payload (Refer to clause 9.2.1 for trigger payload description).

Step 2.2 Check the *notificationEventCat* attribute:

- If the *notificationEventCat* attribute is set, the Notify request primitive shall employ the ***Event Category*** parameter as given in the *notificationEventCat* attribute. Then continue with the step 2.3.

- If the *notificationEventCat* attribute is not configured, then continue with step 2.3.

Step 2.3 Check the *latestNotify* attribute:

- If the *latestNotify* attribute is set, the Originator shall assign ***Event Category*** parameter of value 'latest' of the notifications generated pertaining to the subscription created.

Step 2.4 Check the batching notifications policy:

- See details in oneM2M TS-0001 [6], clause 10.2.10.7.

NOTE: The use of some attributes such as *rateLimit* and *preSubscriptionNotify* is not supported in the present document.

Step 2.5 Check the *notificationURI* attribute:

- The Originator shall fetch the *notificationURI* attribute and set the value to the ***To*** parameter of the Notify request. When the *notificationURI* attribute contains more than one target, the Originator shall generate each Notify request per target.

- If the *notificationURI* attribute includes the notification serialization indication, in form of key-value pair, e.g. "ct=json", after the delimiter "?", the Originator shall serialize the notification for the notification target in that serialization type. The delimiter with the serialization indication shall be removed when the target is set to the ***To*** parameter of the Notify request. Then continue with step 3.0.

Step 3.0 The Originator shall check the notification and reachability schedules, but the notification schedules may be checked in different order:

- If the <subscription> resource associated with the modified resource includes a <notificationSchedule> child resource, the Originator shall check the time periods given in the *scheduleElement* attribute of the <*notificationSchedule*> child resource.

- Also, the Originator shall check the reachability schedule associated with the Receiver by exploring its <schedule> resource. If reachability schedules are not present in a Node then that Entity is considered to be always reachable.

- If notificationSchedule and reachability schedule indicate that message transmission is allowed, then proceed with step 5.0. Otherwise, proceed with step 4.0.

- In particular, if the *notificationEventCat* attribute is set to 'immediate' and the <notificationSchedule> resource does not allow transmission, then go to step 5.0 and send the corresponding Notify request primitive by temporarily ignoring the Originator's notification schedule.

Step 4.0 Check the *pendingNotification* attribute:

- If the *pendingNotification* attribute is set, then the Originator shall cache pending Notify request primitives according to the *pendingNotification* attribute. The possible values are 'sendLatest' and 'sendAllPending'. If the value of pendingNotification is set to 'sendLatest', the most recent Notify request primitive shall be cached by the Originator and it shall set the ***Event Category*** parameter to 'latest'. If *pendingNotification* is set to 'sendAllPending', all Notify request primitives shall be cached by the Originator. If the *pendingNotification* attribute is not configured, the Originator shall discard the corresponding Notify request primitive. The processed Notify request primitive by the *pendingNotification* attribute is sent to the Receiver once message transmission becomes possible (see the step 6.0).

Step 5.0 Check the *expirationCounter* attribute:

- If the *expirationCounter* attribute is set, then it shall be decreased by one when the Originator successfully sends the Notify request primitive. If the counter equals to zero('0'), the corresponding <subscription> resource shall be deleted. Then end the 'Compose Notify Request Primitive' procedure.

- If the *expirationCounter* attribute is not configured, then end the 'Compose Notify Request Primitive' procedure.

When message transmission becomes possible, the Originator shall execute the following steps in order:

Step 6.0 If the *pendingNotification* attribute is set, the Originator shall send the processed Notify request primitive by the *pendingNotification* attribute and then continue with the step 7.0

Step 7.0 Check the *expirationCounter* attribute:

- If the *expirationCounter* attribute is set, then its value shall be decreased by one when the Originator successfully sends the Notify request primitive. If the counter meets zero, the corresponding <subscription> resource shall be deleted. Then end the 'Compose Notify Request Primitive' procedure.

- If the *expirationCounter* attribute is not configured, then end the 'Compose Notify Request Primitive' procedure.

Receiver:

When the Hosting CSE receives a Notify request primitive, the Hosting CSE shall check validity of the primitive parameters. In case the Receiver is a transit CSE which forwards or aggregates Notify request primitives before sending to the subscriber or other transit CSEs, upon receiving the Notify request primitive with the ***Event Category*** parameter set to 'latest', the Receiver shall identify the latest Notify request primitive with the same subscription reference while storing Notify request primitives locally. When the Receiver as a transit CSE needs to send pending Notify request primitives, it shall send the latest Notify request primitive. When the Receiver as a transit CSE needs to send Notify request primitives, it shall use one of the serializations specified in the subscriber or other transit CSE *contentSerialization* attribute. If there is no *contentSerialization* value specified the transit CSE may use any serialization format.

7.5.1.2.3 Subscription Verification during Subscription Creation

Originator:

When the Originator is triggered to perform subscription verification (clause 7.4.8.2.1) during <subscription> creation procedure, it performs the following steps in order:

1) Set the *verificationRequest* element of the notification data object as true in the Notify request primitive.

2) Set the *creator* element of the notification data object as the Originator ID of the <subscription> creation in the primitive.

3) Set the ***To*** parameter as the *notificationURI* in the primitive. If the *notificationURI* contains more than one value, then set the other value to the duplicated primitives from step 2).

4) Send the Notify request primitive(s).

Receiver:

When the Hosting CSE receives a Notify request primitive which includes ***verificationRequest*** element of the notification data object set as true, the Hosting CSE shall check if the creator and the Originator have NOTIFY privilege to the *notificationURI*.

If it fails, the Hosting CSE shall return a ***Response Status Code*** indicating "SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE" or "SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE" error, respectively, with the Notify response primitive. Otherwise, it shall return successful response primitive.

7.5.1.2.4 Notification of Subscription Deletion

Originator:

When the <subscription> resource is deleted the Originator shall send Notify request primitives to the <subscription> resource's *subscriberURI* and *associatedCrossResourceSub* if they are configured:

a) subscriptionDeletion element of the notification data object set as true.

b) subscriptionReference element of the notification data object set as the resource identifier of the <subscription> resource.

c) The ***To*** parameter shall be set to the entity indicated in *subscriberURI* or *associatedCrossResourceSub*.

Receiver:

When a Hosting CSE receives a subscription deletion notification targeted to an existing <crossResourceSubscription> resource then the <crossResourceSubscription> resource is deleted using the procedure defined in clause 7.4.58.2.4.

7.5.1.2.5 Notification for Asynchronous Non-blocking Request

Originator:

When the requested operation for a nonBlockingRequestAsynch request is completed, the Originator (=Hosting CSE of the resource) shall send a Notify request primitive to inform the final result of requested operation against the oneM2M resource. When the notificationURI was present and empty in the ***Response Type*** parameter in the previously received nonBlockingRequestAsynch request, no notification with the result of the requested operation shall be sent at all by the Originator. Otherwise, the Originator shall send a Notify request primitive as follows: (If the notificationURI was present and contains multiple entries, then the Originator shall send a Notify request primitive for each entry in the notificationURI list.)

a) The ***From*** parameter shall be set to the ID of the Originator (i.e. Hosting CSE which hosts the resource targeted by the previously received nonBlockingRequestAsynch request).

b) If the notificationURI was not present in the ***Response Type*** parameter in the previously received nonBlockingRequestAsynch request, then the ***To*** parameter shall be set to the Originator of the previously received nonBlockingRequestAsynch request.
If the notificationURI was present and not empty in the ***Response Type*** parameter in the previously received nonBlockingRequestAsynch request, then the ***To*** parameter shall be set to the next notificationURI list entry.

c) The ***Response Type***: If the Originator chooses to send the Notification in nonBlockingRequestAsynch mode, the Originator shall include a notificationURI in the ***Response Type*** and set it to empty.

d) The ***Content*** parameter shall be set to the response to the previously received nonBlockingRequestAsynch request as m2m:responsePrimitive.

Receiver:

No change from the generic procedure in clause 7.2.2.2.

7.5.1.2.6 Notification for subscription via group

Whenever the subscription relationship is established through a <group> resource and either:

1) modification of a subscribed-to resource triggers a notification procedure as defined in clause 7.5.1.2.2, or

2) the subscribed-to sub-<group> triggers a aggregated notification procedure as defined in clause 7.5.1.2.6, the following procedures shall be performed.

The Member hosting CSE shall perform the steps defined in clause 7.5.1.2.2.

The Group hosting CSE shall perform the following steps in order:

1) Validate if the notification or the aggregated notification has been sent from one of its own member resources when it gets a notification at the notificationURI. The group hosting CSE shall return a response primitive with the ***Response Status Code*** indicating "ORIGINATOR_HAS_NO_PRIVILEGE" error if the validation fails.

2) Upon successful validation, the group hosting CSE shall collect notification requests targeted at the same subscriber according to the notificationForwardingURI element of each notification data object. The group hosting CSE shall aggregate the notification requests into an aggregatedNotification element of the notification data object. If the group hosting CSE receives an aggregated notification from a group member then it shall extract the notifications contained in that aggregated notification and insert them into the aggregatedNotification element (aggregatedNotification(s) are not nested). If the duration is not specified, then the Hosting CSE shall batch notifications using the default duration value as given by the M2M Service Provider. When the group Hosting CSE gets first notification, it starts new aggregation by the number of notifications and/or the duration in notifyAggregation attribute.

3) When the number or duration time is reached, send the aggregated notification to the *notificationURI* according to the *notificationForwardingURI* element in the notification data object. Then the group Hosting CSE waits for next notification to start aggregation. In case the group hosting CSE is member of another group hosting CSE through which the subscription is created, the notification request shall be sent according to the mapping of the *notificationURI* of the two group hosting CSEs. When aggregating the notification requests, the group hosting CSE may utilize the ***Request Expiration Timestamp*** parameter of the notification request primitive to determine the time by which the aggregated notifications need to be sent.

4) "Wait for Response primitive" procedure.

5) Upon receiving the response, the group hosting CSE shall send the response separately to each individual member hosting CSEs to respond their corresponding notify request.

The group hosting CSE shall perform notification aggregation while the <group> resource has not expired and has a *notificationForwardingURI* attribute value.

The Subscriber shall perform the following steps in order:

1) Extract each notification from the aggregated notification.

2) Treat the notification as if it is sent from the original subscribed-to resource.

3) "Create a success response" procedure.

4) "Send the Response primitive" procedure.

7.5.1.2.7 Notification for service layer long polling

When an AE or CSE receives a long polling response (clause 7.4.22.2.2 Retrieve <pollingChannelURI>) containing a request primitive, then the AE or CSE shall generate an appropriate response for the request in the long polling response. The AE or CSE, as the Originator, shall send the response in a Notify request to the <pollingChannelURI> Hosting CSE.

The Originator shall compose a Notify Request primitive according to clause 7.3.1.1 with following additional parameter settings:

1) The ***To*** parameter shall be set to the address of the <pollingChannelURI> that belongs to the Originator.

2) The ***Content*** parameter shall be set to the response to the previously received request as a m2m:responsePrimitive.

3) The ***Operation*** parameter shall be set to NOTIFY.

7.5.1.2.8 Notification for on-demand discovery request

In this procedure, the Originator is the Hosting CSE which performs resource discovery and the Receiver is the IPE.

Originator:

The Hosting CSE shall include the Originator ID of the original discovery request and the ***Filter Criteria*** in the Notify request. The ***To*** parameter of this request shall be set to the *pointOfAccess* of the <AE> resource of the IPE.

Receiver:

When the IPE receives the Notify request, it shall check the Originator ID and determines whether it performs external discovery. If the IPE does not accept the discovery, it shall send the unsuccessful response with the ***Response Status Code*** indicating "DISCOVERY_DENIED_BY_IPE". If the IPE accepts the discovery, it performs external discovery. If the discovery result contains one or more match, the IPE shall create resource(s) for the result and send the Notify response including the list of created resource(s) in the ***Content*** parameter. The IPE may configure "Discover" privilege for the original discovery request Originator for the newly created resource(s).

7.5.1.2.9 Notification for missing Time Series Data

When an AE wants to be informed of the number of missing data points in a given renewable time duration, the AE should request the creation of a <*subscription*> resource and set the *missingData* in the *eventNotificationCriteria* conditions to specify the reporting policy. This enables the AE to keep track of the number of missing data points and the corresponding time-stamps over a predefined but renewable duration (i.e. the "duration" of the *missingData*).

Originator(Hosting CSE):

No change from the procedures in clause 7.2.2.1 except the following addition in Step1.0:

When the first missing data point is detected (i.e. a detection of the first discontinuous time-stamp), following the creation of the subscription, the Hosting CSE shall start a timer, and keep counting the number of the missing data

points. The timer is set according to the *duration* in the *missingData*. The reporting policy is governed by the rules below:

- If the total number of missing data points becomes equal to or greater than the *number* specified in *missingData* condition before the timer expires, a Notify request shall be sent with the *missingDataList* and *currentMissingDataNr* included in the Notify request. The missing data points counter shall continue counting while the timer continues to run (since it did not expire). Initiating Notify request to report missing data points shall follow the same logic described above until the timer expires (see next bullet for behaviour when the timer expires).
- If the timer expires, the timer is restarted, and the missing data points counter is reset back to 0.
- The renewal of the timer and the missing data points counter upon timer expiry shall continue until such time as the subscription is cancelled or terminated. Once a subscription is terminated, a final Notify request is sent out with the current number of missing data points and the timer is stopped.
- If no missing data points have been detected at all during the life time of a subscription, then no timer shall be started at all. But once a timer is started triggered by the first missing data point, then the above rules in the previous bullets shall apply.

No change for the remaining steps from the procedures in clause 7.5.1.2.2.

7.5.1.2.10 Notification for Dynamic Authorization

Dynamic Authorization uses a two-way exchange of information between a Hosting CSE (i.e. Originator) and a Dynamic Authorization System (DAS) Server (i.e. Receiver). This two-way handshake is performed using a Notification request and response.

Originator:

When the Originator (i.e. Hosting CSE) is triggered to perform dynamic authorization for an incoming request that it receives, then it performs the following steps in order:

1) Configure the ***To*** parameter with the address of the corresponding DAS Server associated with the resource targeted by the received request. The Hosting CSE shall use the DAS Server address information configured within the *dynamicAuthorizationPoA* attribute of the <*dynamicAuthorizationConsultation*> resource associated with the targeted resource. The Hosting CSE shall determine the corresponding <*dynamicAuthorizationConsultation*> resource using the *dynamicAuthorizationConsultationIDs* attribute of the targeted resource. If the attribute is not supported by the targeted resource, or it is not set, or it has a value that does not correspond to a valid <*dynamicAuthorizationConsultation*> resource(s), or it refers to a <*dynamicAuthorizationConsultation*> resource(s) that is not reachable, then based on system policies, the *dynamicAuthorizationConsultationIDs* associated with the parent may apply to the child resource if present, or a system default <*dynamicAuthorizationConsultation*> may apply if present. If a *dynamicAuthorizationConsultationID* attribute and corresponding <*dynamicAuthorizationConsultation*> resource cannot be found or if the *dynamicAuthorizationEnabled* of a <*dynamicAuthorizationConsultation*> has a value of false, the Hosting CSE shall reject the request by returning an "ORIGINATOR_HAS_NO_PRIVILEGE" ***Response Status Code*** to the Originator of the received request and no additional steps shall be performed.

2) Configure the ***From*** parameter with the ID of the Hosting CSE which hosts the resource targeted by the received request.

3) Configure the mandatory sub-elements of the *securityInfo* element of the notification data:

   a) The *securityInfoType* element shall be configured as "1" (Dynamic Authorization Request) in the Notify request primitive.

   b) The *originator* element shall be configured with the ID of the Originator of the received request.

   c) The *targetedResourceType* element shall be configured with the type of resource targeted by the received request.

   d) The *operation* element shall be configured with the type of operation targeted by the received request.

4) Optionally configure one or more optional sub-elements of the *securityInfo* element of the notification data:

   a) The *originatorIP* element may be configured with the IP address of the Originator of the received request.

   b) The *originatorLocation* may be configured with the location of the Originator of the received request.

   c) The *originatorRole* may be configured with the role of the Originator of the received request.

   d) The *requestTimestamp* may be configured with the time at which the request was received.

   e) The *targetedResourceID* may be configured with the identifier of the targeted resource of the received request.

   f) The *proposedPrivilegesLifetime* may be configured with a time duration for which the Hosting CSE is requesting privileges be granted to the Originator of the received request.

   g) The *rolesFromACPs* may be configured with a list of roles specified in the ACPs associated with the resource targeted by the received request.

   h) The *tokenIDs* may be configured with a list of token identifiers specified by the Originator of the received request.

   i) The *authorSignIndicator* may be configured with the value specified by the received request.

5) The Hosting CSE shall send the notification request for dynamic authorization to the targeted DAS Server.

Receiver:

When the DAS Server receives a notification request for dynamic authorization, it processes the request and returns a notification response for dynamic authorization to the originating Hosting CSE.

NOTE 1: The details of how the DAS Server processes the notification request for dynamic authorization are not visible to the oneM2M system, and are not addressed in the present document.

Originator:

When the Hosting CSE receives a notification response for dynamic authorization, it performs the following steps in order:

1) The Hosting CSE shall verify that the *securityInfoType* element of the *securityInfo* element of the notification is configured as "2" (Dynamic Authorization Response). If it is not, the Hosting CSE shall not grant privileges to the Originator of the request for which the Hosting CSE was attempting dynamic authorization. The Hosting CSE shall reject the request by returning an "ORIGINATOR_HAS_NO_PRIVILEGE" ***Response Status Code*** to the Originator of the received request and no additional steps shall be performed.

2) The Hosting CSE shall check whether the response contains a *dynamicACPInfo* element. If present, the Hosting CSE shall create a <*accessControlPolicy*> child resource under the targeted resource and configure its privileges using the *dynamicACPInfo*. In this case, the Hosting CSE shall configure the *privileges* attribute with the *grantedPrivileges* and the *expirationTime* attribute with the *privilegesLifetime*. The Hosting CSE shall also configure the *selfPrivileges* attribute to allow itself to perform Update/Retrieve/Delete operations on the newly created <*accessControlPolicy*> resource.

3) The Hosting CSE shall check whether the response contains a *tokens* element. If present the Hosting CSE shall perform verification and caching of the token as specified in clause 7.3.2 in oneM2M TS-0003 [7] and the Hosting CSE shall check whether an *AuthorSignReqInfo* is contained in the response. If it is present, the Hosting CSE shall add the *AuthorSignReqInfo* into the response to the Originator of the incoming request.

NOTE 2: The Hosting CSE uses the information in the DAS response for authorization, see clause 7.3.3.15.

7.5.1.2.11 Notification for receiverESPrimRandObject Generation

Originator:

When the Originator is triggered to perform receiverESPrimRandObject generation as part of establishing sessionESPrimKey with a Receiver, it performs the following steps in order:

1) Configure the ***To*** parameter with the address of the Receiver's <CSEBase> or <AE> resource.

2) Configure the ***From*** parameter with the ID of the Originator.

3) The securityInfoType element of the securityInfo element of the notification data shall be configured as "3" (receiverESPrimRandObject Request) in the Notify request primitive.

4) The Originator shall send the notification request for dynamic authorization to the targeted Receiver.

Receiver:

When a Receiver receives a notification request, it processes the securityInfoType element of the securityInfo element and determines that the notification is for receiverESPrimRandObject generation. The Receiver generates a receiverESPrimRandObject as specified in oneM2M TS-0003 [7]. Then the Receiver sends a notification response to the Originator, with the securityInfo element containing the following elements:

- The securityInfoType element shall be configured as "4" (receiverESPrimRandObject Response) in the Notify response primitive.
- The esprimRandObject shall contain the receiverESPrimRandObject.

When the Originator receives a notification response for dynamic authorization, it performs the following steps in order:

1) The Originator shall verify that the securityInfoType element of the securityInfo element of the notification is as configured as "4" (receiverESPrimRandObject Response).

2) The Originator shall check whether the securityInfo element contains an esprimRandObject element. If it does, the Originator shall use the receiverESPrimRandObject in esprimRandObject element in the generation of sessionESPrimKey.

7.5.1.2.12 Notification for End-to-End Security Certificate-based Key Establishment (ESCertKE)

The End-to-End Security Certificate-based Key Establishment (ESCertKE) uses a four-way exchange of messages to establish a symmetric key for end-to-end security between the Originator and Receiver. This four-way exchange consists of two sequential two-way (request and response) exchanges using Notification for transport.

Originator:

When the Originator is to perform ESCertKE with the Receiver, then the Originator performs following steps in order:

1) The Originator shall form the ESCertKE Message 1 as specified in oneM2M TS-0003 [7].

2) The Originator performs the general procedure for an Originator as described in clause 7.2.2.1 with the following additional details:

   a) In step Orig-1.0, described in clause 7.3.1.1, a Notify Request primitive shall be formed with the Notification data comprising a securityInfo element with securityInfoType element set to "6" (ESCertKE Message) and escertkeMessage element set to the value of the ESCertKE message 1.

Receiver:

When a Receiver receives the Request, then the Receiver performs the general procedure for a Receiver with the following additional steps performed as part of Recv-6.5 for Notify Retargeting described in clause 7.3.3.9:

1) The Receiver extracts the securityInfo element from the notification data.

2) The Receiver examines the securityInfoType element of the securityInfo element and determines from its value "6" (ESCertKE Message) that the notification is for ESCertKE.

3) The Receiver extracts the ESCertKE message 1 contained in escertkeMessage element of the securityInfo element.

4) The Receiver processes ESCertKE message 1 as specified in clause 8.7 of oneM2M TS-0003 [7], resulting in ESCertKEY message 2.

5) The Receiver forms the Response content comprising a securityInfo element with securityInfoType element set to "6" (ESCertKE Message) and escertkeMessage element set to the value of the ESCertKE message 2.

Originator:

The Originator performs the following steps at Orig-6.0 "Process Response Primitive":

1) The Originator extracts the securityInfo element from the Response ***Content*** parameter.

2) The Originator examines the securityInfoType element of the securityInfo element and verifies that from its value is "6" (ESCertKE Message).

3) The Originator extracts the ESCertKE message 2 contained in escertkeMessage element of the securityInfo element.

4) The Originator processes ESCertKE message 2 as specified in clause 8.7 of oneM2M TS-0003 [7], resulting in ESCertKEY message 3.

The Originator and Receiver now repeat steps 2) to 10), with ESCertKE message 3 replacing ESCertKE message 1, and ESCertKE message 4 replacing ESCertKE message 2. Following the repeat of Steps 2) to 10), the following step is performed to conclude the procedure:

1) The Originator processes ESCertKE message 2 as specified in clause 8.7 of oneM2M TS-0003 [7].

7.5.1.2.13 Using Notify for transport of ESPrim Objects

The Notify operation is used for transport of ESPrim Objects protecting an exchange of inner primitives, as part of clause 7.6.2 "Procedure for applying End-to-End Security of Primitives (ESPrim)".

NOTE: The inner request primitive can request any operation (Create, Retrieve, Update, Delete, Notify) allowed by the resource addressed by the inner request primitive. The composing or processing of the inner request primitive and the corresponding inner response primitive follows the general procedures in clause 7.2.2, and is not addressed further in the present clause.

Originator:

When the originator is to secure an exchange of inner primitives, then the Receiver shall apply the following steps in order:

1) Encrypt the inner request primitive to form an ESPrim Object as described in step E of clause 7.6.2.

2) Form the Notify request from the ESPrim Object as described in step F of clause 7.6.2.

3) Deliver the Notify request to the Receiver as described in step G and H of clause 7.6.2.

Receiver:

When a Receiver receives a Notify request transporting an ESPrim Object, then the Receiver shall apply the following steps in order:

1) Process the Notify request to extract the ESPrim Object as described in step I of clause 7.6.2.

2) Decrypt the ESPrim Object to form the inner request primitive as described in step J of clause 7.6.2.

3) The inner request primitive is processed, resulting in an inner response primitive.

4) Encrypt the inner response primitive to form an ESPrim Object as described in step L of clause 7.6.2.

5) Form a successful Notify response as described in step M of clause 7.6.2.

6) Deliver the Notify request to the Originator as described in step N of clause 7.6.2.

Originator:

When the Originator receives a Notify response transporting an ESPrim Object, then the Receiver shall perform the following steps in order:

1) Process the Notify request to extract the ESPrim Object as described in step O of clause 7.6.2.

2) Decrypt the ESPrim Object to form the inner request primitive as described in step P of clause 7.6.2.

3) The inner response primitive is processed.

Error cases are addressed in clause 7.6.2.

7.5.1.2.14 Notification for Authorization Relationship Mapping Record creation

Originator-AE:

The AE sends the AuthorSigns to the Hosting CSE using a Notification Request. It performs the following steps:

1) Configure the ***To*** parameter with the ID of the Hosting CSE.

2) Configure the ***From*** parameter with the ID of the Originator-AE.

3) Configure the mandatory sub-elements of the *securityInfo* element of the notification data:

   a) The *securityInfoType* element shall be configured as "7" (Dynamic Authorization Relationship Mapping Request) in the Notify request primitive.

4) Optionally configure one or more optional sub-elements of the *securityInfo* element of the notification data:

   a) The *authorSigns* element shall be configured with the value of the *authorSigns* generated by the Originator-AE.

   b) The *tokenIDs* element shall be configured with the a list of token identifiers.

   c) The *tokens* element may be configured with a list of tokens.

NOTE 1: At least one of the Token representation elements (*tokenIDs* and *tokens*) shall be present in the request.

Receiver/Originator-the Hosting CSE:

When the Hosting CSE receives a request including *authorSigns* from AE, it will forward the *authorSigns* to the DAS Server AE using a Notification Request:

1) Configure the ***To*** parameter:

   No change from the procedures in clause 7.5.1.2.10.

2) Configure the ***From*** parameter:

   No change from the procedures in clause 7.5.1.2.10.

3) Configure the mandatory sub-elements of the *securityInfo* element of the notification data:

   a) The *securityInfoType* element shall be configured as "7" (Dynamic Authorization Relationship Mapping Request) in the Notify request primitive.

   b) The *originator* element shall be configured with the ID of the Originator-AE of the received request.

4) Optionally configure one or more optional sub-elements of the *securityInfo* element of the notification data:

   a) The *authorSigns* element shall be configured with the value of the *authorSigns* specified by the Originator-AE of the received request.

   b) The *tokenIDs* element shall be configured with the a list of token identifiers specified by the Originator-AE of the received request.

   c) The *tokens* element may be configured with a list of tokens.

NOTE 2: At least one of the Token representation elements (*tokenIDs* and *tokens*) shall be present in the request.

Receiver:

When the DAS server receives a notification to create an Authorization Relationship Mapping Record, it processes the request and returns a notification response to the originating Hosting CSE.

NOTE 3: The details of how the DAS Server processes the notification request for dynamic authorization are not visible to the oneM2M system, and are not addressed in the present document.

7.5.1.2.15 Notification for Authorization Relationship Mapping Record Request

Originator:

When the Hosting CSE receives an incoming request from an AE including ***Tokens*** or ***TokenIDs*** which are valid but the holder attribute stored locally for the Token(s) is not equal to the AE-ID which sent the incoming request, the Hosting CSE shall send a Notification request to DAS server AE to get the value of *AuthorSigns* for this Token:

1) Configure the ***To*** parameter:

   No change from the procedures in clause 7.5.1.2.10.

2) Configure the ***From*** parameter:

   No change from the procedures in clause 7.5.1.2.10.

3) Configure the mandatory sub-elements of the *securityInfo* element of the notification data:

   a) The *securityInfoType* element shall be configured as "7" (Dynamic Authorization Relationship Mapping Request) in the Notify request primitive.

4) Optionally configure one or more optional sub-elements of the *securityInfo* element of the notification data:

   a) The *tokenIDs* element shall be configured with the a list of token identifiers specified by the Originator-AE of the received request.

   b) The *tokens* shall be configured with a list of tokens specified by the Originator of the received request.

   c) The a*uthorSignReqInfo* element shall be configured with the value true to request the Signature for the token which is contained in the Authorization Relationship Mapping Record.

NOTE 1: At least one of the Token representation elements (*tokenIDs* and *tokens*) shall be present in the request.

Receiver:

When the DAS server receives the Notification request for requesting the signature, it processes the request and returns a notification response including the Signature to the originating Hosting CSE.

NOTE 2: The details of how the DAS Server processes the notification request for dynamic authorization are not visible to the oneM2M system, and are not addressed in the present document.

Originator:

When the Hosting CSE receives a notification response for Authorization Relationship Mapping, it performs the following steps in order:

1) The Hosting CSE shall verify that the *securityInfoType* element of the *securityInfo* element of the notification is configured as "8" (Dynamic Authorization Relationship Mapping Response).

2) The Hosting CSE shall check if the *Signature* element is in the response. If the *Signature* element is present, the Hosting CSE shall further confirm whether the AE has the possession of the token sent in the incoming request.

7.5.1.2.16 Notification of a Change in AE Registration Point

Originator:

Case a) When an AE that wants its registration points tracked, attempts to re-register to a new Registrar CSE using an AE-ID-Stem starting with a 'C', that was previously assigned by a prior Registrar CSE, the new Registrar CSE, acting as the Originator, shall send a Notify request primitive to inform the IN-CSE that the AE has changed registration point.

Case b) When the IN-CSE determines that an AE has changed its registration point, the IN-CSE, acting as the Originator, shall send a Notify request primitive to the CSEs, so that these may update the references to the <AE> resources for the AE that has changed its registration point. If the IN-CSE maintains an <AEContactList> resource, the IN-CSE shall determine which CSEs are effected, and shall send a Notify request primitive only to these. If the IN-CSE does not maintain an <AEContactList> resource, the IN-CSE shall send a Notify request primitive to all CSEs.

In both cases, the Originator shall:

1) Set the *AERegistrationPointChange* element of the notification data object as true in the Notify request primitive.

2) Set the *trackingID1* element of the notification data object as SP-relative-Resource-ID at the prior registration point.

3) Set the *trackingID2* element of the notification data object as SP-relative-Resource-ID at the new registration point.

4) Send the Notify request primitive(s).

Receiver:

For both cases, upon receiving the Notify request primitive with an *AERegistrationPointChange* element of the notification data object set as true, the Hosting CSE shall use this to indicate a change in registration point. The Receiver shall:

1) Update references to the SP-Relative-Resource-ID (e.g. in Announce links, Notification targets, group Member IDs, <accessControlPolicy> resource OriginatorID lists) tied to the prior registration point (trackingID1), so that these refer to the new registration point (trackingID2).

2) Update the status of this registration to "INACTIVE". if the Receiver hosts the registration of the prior AE registration point.

7.5.1.2.17 Notification of a Reference to Application Entity Resource identifier

Originator:

When a Hosting CSE (acting as Originator) determines that a child resource has a modified (new, updated, or deleted) reference an Application Entity Resource ID, the Hosting CSE shall send a Notify request primitive to the IN-CSE, requesting to add, update, or delete the entry to the <AEContactList> resource. The Originator shall:

1) Set the *AEReferenceIDChange* element of the notification data object as true in the Notify request primitive.

2) For a new AE reference (from a CREATE operation):

   a) Set *trackingID1* element of the notification data object to NULL.

   b) Set *trackingID2* element of the notification data object to the SP-relative-Resource-ID used in the new AE reference.

3) For an updated AE reference (from an UPDATE operation):

   a) Set *trackingID1* element of the notification data object to the SP-relative-Resource-ID used in the AE reference before the UPDATE.

   b) Set *trackingID2* element of the notification data object to the SP-relative-Resource-ID used in the AE reference after the UPDATE.

4) For a deleted AE reference (from an DELETE operation or resource expiration):

   a) Set *trackingID1* element of the notification data object to the SP-relative-Resource-ID used in the AE reference.

   b) Set *trackingID2* element of the notification data object to NULL.

5) Send the Notify request primitive(s).

Receiver:

Upon receiving the Notify request primitive with an *AEReferenceIDChange* element of the notification data object set as true, the IN-CSE (acting as Receiver) shall update its <AEContactList> resource. IN-CSE will add, update, or remove entries from the *AE-IDList* attribute of the <AEContactListPerCSE> resource corresponding to the impacted CSE.

7.5.1.2.18 Cross-Resource Notification

When the <crossResourceSubscription> Hosting CSE receives a notification from the Host of a <subscription> indicated in *regularResourcesAsTarget* or *subscriptionResourcesAsTarget* the <crossResourceSubscription> Hosting CSE shall perform the following steps:

1) The Hosting CSE shall send a notification response to the <subscription> resource Hosting CSE.

2) Aggregate notifications using the time window mechanism indicated by *timeWindowType* attribute of the <crossResourceSubscription> resource to determine if a cross-resource notification shall be issued:

   a) The Hosting CSE shall store the received notification until the current time window expires. When the current time window expires, the Hosting CSE shall discard stored notifications:

      i) If *timeWindowType* is PERIODICWINDOW then a new time window shall be started when the current time window expires.

      ii) If *timeWindowType* is SLIDINGWINDOW then a new time window shall be started when the next notification is received.

   b) When notifications from all target <subscription> resources occur within the required time window the Hosting CSE shall issue a cross-resource notification in a notification data object with type m2m:notification with *subscriptionReference* element set as the URI of the <crossResourceSubscription> resource.

3) Send the notification to the notificationURI using the procedure defined in clause 7.5.1.2.2.

4) "Wait for Response primitive" procedure.

The **Subscriber or Notification Targets** which receive cross-resource notifications from the Hosting CSE shall perform the following steps in order:

1) "Create a success response" procedure defined in clause 7.3.3.12.

2) "Send the Response primitive" procedure.

7.5.1.2.19 Notification for Subscription Blocking Triggered update

Whenever the Hosting CSE receives an update request primitive for a target resource which has subscription with *notificationEventType* set to "Blocking_Update", it shall perform the steps listed below before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation" is performed.

1) If the *attribute* condition tag of the *eventNotificationCriteria* attribute of the <subscription> resource is set, check that the *attribute* condition tag matches the modified attributes in the received UPDATE request.

2) Prevent or block all other UPDATE request primitives to this target resource.

3) Create a Notification request primitive and configure the request parameters as follows.

   a) Set the *representation* attribute of the notification to the representation of the target resource contained in the received UPDATE request primitive.

4) Send the Notification request primitive to the target specified in *notificationURI*.

5) Wait for a Notification response.

6) Process the Notification response primitive

a) If the notification ***Response Status Code*** is not successful, return a response to the original blocked UPDATE request primitive with a ***Response Status Code*** indicating NOT_ACCEPTABLE.

b) If the notification ***Response Status Code*** is successful, perform Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation" for the received UPDATE request.

7) Allow all other UPDATE request primitives for this target resource.

7.5.2 Elements contained in the Content primitive parameter

Clauses 7.2.1.1 and 7.2.1.2 enumerate the forms that the ***Content*** primitive parameter takes in various Request and Response cases. Note that the ***Content*** primitive parameter is denoted as primitiveContent in both CDT-requestPrimitive-v3_11_0.xsd and CDT-responsePrimitive-v3_11_0.xsd.

This clause details the Objects (elements) used in some of these cases. in the tables below.

The following elements are defined for use in the ***Content*** parameter of a request:

Table 7.5.2-1: Elements used for request content

Element Name	Applicable Operations	Data Type	Defined in
m2m:<resourceType> {other namespace identifier}:<resourceType>	C U	See element declaration	CDT-<resourceType>-v3_11_0.xsd
m2m:notification	N	m2m:notification	CDT-notification-v3_11_0.xsd
m2m:aggregatedNotification	N	m2m:aggregatedNotification	CDT-notification-v3_11_0.xsd
m2m:securityInfo	N	m2m:securityInfo	CDT-notification-v3_11_0.xsd
m2m:attributeList	R	m2m:attributeList	CDT-requestPrimitive-v3_11_0.xsd
m2m:responsePrimitive	N	Anonymous data type defined in the responsePrimitive declaration	CDT-responsePrimitive-v3_11_0.xsd

The following elements are defined for use in the ***Content*** parameter of a response sent in reply to a request message with ***Operation*** and ***Result Content*** (rcn) parameters as given in the column "Applicable Operations" (the settings of the ***Result Content*** parameters are defined in clause 6.3.4.2.7; NP means the rcn parameter is not present).

Table 7.5.2-2: Elements used for response content

Element Name	Applicable Operations/rcn	Data Type	Element is Defined in
m2m:<resourceType> {other namespace identifier}:<resourceType> See note 6	C/1,9,NP R/1,4,5,6,7,8,NP U/1,9,NP D/1,4,5,6,8 See note 1	See element declaration	CDT-<resourceType>-v3_11_0.xsd
m2m:resource	C/3	m2m:resourceWrapper	CDT-responsePrimitive-v3_11_0.xsd
m2m:URIList	R/NP See note 2	list of xs:anyURI	CDT-responsePrimitive-v3_11_0.xsd
m2m:resourceRefList	R/6 See note 2	m2m:listOfChildResourceRef	CDT-responsePrimitive-v3_11_0.xsd
m2m:aggregatedResponse	C R U D See note 3	m2m:aggregatedResponse	CDT-responsePrimitive-v3_11_0.xsd
m2m:URI	C/2 See note 4	xs:anyURI	CDT-responsePrimitive-v3_11_0.xsd
m2m:requestPrimitive	See note 7	Anonymous data type defined in the requestPrimitive declaration	CDT-requestPrimitive-v3_11_0.xsd
m2m:debugInfo	See note 5	xs:string	CDT-responsePrimitive-v3_11_0.xsd
m2m:securityInfo	N/NP	m2m:securityInfo	CDT-notification-v3_11_0.xsd
m2m:queryResult	R/10 See note 8	xs:string	CDT-responsePrimitive-v3_11_0.xsd

NOTE 1: The case rcn = 7 applies to Retrieve operation only (R/7). It retrieves the original resource in case the ***To*** parameter points to an announced resource. The case R/NP applies to Retrieve operation (Non-Discovery) only.
NOTE 2: This applies to discovery operation only. For discovery, the format of the address (structured, unstructured) depends on the ***Desired Identifier Result Type*** parameter setting (see clause 6.3.4.2.8).
NOTE 3: This applies to CRUD operations on a <fanOutPoint> child resource of a <group> parent resource. The ***Content*** parameter of each response primitive included in aggregatedResponse is set as given in one of the other rows of this table.
NOTE 4: This also applies to the response ("acknowledgement") to non-blocking requests in asynchronous and synchronous modes for any CRUD operation.
NOTE 5: This is a plain text messages which can optionally be included as debugging information in error responses. The language and content of the message is determined by the Service Provider.
NOTE 6: "{other namespace identifier}" refers to a namespace other than m2m.
NOTE 7: This applies to a polling response that contains a request for polling mechanism (see clause 7.4.22.2.2).
NOTE 8: This applies to semantic query operation only. The Originator may use the Accept option to indicate which media types are acceptable for the semantic query result, e.g. application/sparql-results+xml, or application/sparql-results+json.

The XML schema definition of the ***Content*** primitive parameter (i.e. datatype m2m:primitiveContent) allows to include XML wildcard elements. An XML representation of the ***Content*** primitive parameter shall include a root element which is associated with an XSD Global Element. The root element shall be prefixed with a namespace prefix identifier (e.g. *m2m:*) specified in the associated XSD which defines the respective Global Element. The ***Content*** primitive parameter allows to include namespaces other than m2m.

7.5.3 Multicast group procedures

7.5.3.1 Multicast group CREATE procedure

7.5.3.1.1 Common procedure

The Group Hosting CSE shall perform the following operations:

1) The Group Hosting CSE shall check the *multicastCapability* attribute of each remote Member Hosting CSE's <remoteCSE> resource:

   a) If at least two remote Member Hosting CSEs support multicast capability MBMS, then the Group Hosting CSE shall create 3GPP MBMS multicast group for the members that support MBMS multicast capability, refer to clause 7.5.3.1.2.

b) If at least two remote Member Hosting CSEs support multicast capability IP, then the Group Hosting CSE shall create an IP multicast group for the members that support IP multicast capability, refer to clause 7.5.3.1.3.

Table 7.5.3.1.1-1: Elements of Multicast Group Information Data Object

Element Name	Description
groupID	Indicates the *<group>* resource identifier that the Multicast Group Information Data Object applies to.
multicastType	Indicates the type of multicasting supported by the group members. Different multicast types within one group shall be created as different data objects.
externalGroupID	Used only for 3GPP MBMS multicast groups. Has the same value as the attribute *externalGroupID* of the related <localMulticastGroup> resources, see clause 7.4.56. Different external Group IDs within one multicast type shall be created as different data objects.
multicastAddress	Has the same value as the attribute *multicastAddress* of the related <localMulticastGroup> resources, see clause 7.4.56.
multicastGroupFanoutTarget	Has the same value as the attribute *multicastGroupFanoutTarget* of the related <localMulticastGroup> resources, see clause 7.4.56.
memberList	Contains all the members of the *memberLists* of the related <localMulticastGroup> resources.
groupServiceServerAddress	Used only for 3GPP MBMS multicast group. A group hosting CSE that supports MBMS Multicast will have this value provisioned by the 3GPP service provider. See oneM2M TS-0001 [6], clause 10.2.7.13.
TMGI	Used only for 3GPP MBMS multicast group. Has the same value as the attribute *TMGI* of the related <localMulticastGroup> resources, see clause 7.4.56.
TMGIExpiration	Used only for 3GPP MBMS multicast group. Indicates the duration of the TMGI expiration time. See oneM2M TS-0001 [6], clause 10.2.7.13.
responseTimeWindow	Has the same valued as the attribute *responseTimeWindow* of the related <localMulticastGroup> resources, see clause 7.4.56.

7.5.3.1.2 3GPP™ MBMS group creation procedure

The Group Hosting CSE shall perform the operations which are specified in clause 10.2.7.13.1 of oneM2M TS-0001 [6] and in clause 7.7.3.1 of oneM2M TS-0026 [43]:

1) Get the *externalGroupID* of the <remoteCSE> resource for each member whose *externalGroupID* exists, send a 3GPP Allocate TMGI Request [51] to the group Service Server Address as local configuration over Mcn reference point.

2) When the Group Hosting CSE receives the corresponding response from 3GPP network:

   a) If the response indicates failure, then the Group Hosting CSE shall stop the multicast group create procedure.

   b) If the response indicates success, then the Group Hosting CSE shall get the TMGI and TMGIExpiration from the response.

3) Create a Multicast Group Information data object locally for each distinct *externalGroupID*: set its groupID to the value of the created <group>, assign the multicastType to MBMS, allocate a multicastAddress, set the externalGroupID to the same value as the *externalGroupID* of the <remoteCSE> resources, set the memberList to the members that belong to this multicast group, set the groupServiceServerAddress to the value provisioned by the 3GPP service provider, set the TMGI and TMGIExpiration to the values in the response from 3GPP network, set responseTimeWindow according to local policy. Assign a{fanout-segment} string, and set the multicastGroupFanoutTarget to /{groupHostingCSE-ID}/ {fanout-segment}.

NOTE: For a guide to an allocation scheme of IPv4 and IPv6 multicast address spaces, reference may be made to standard documents such as IETF RFC 5771 [i.10] and IETF RFC 4291 [i.11].

4) Send a Create <localMulticastGroup> primitive to each Member Hosting CSE in the multicast group:

   a) Prepare the attributes of the <localMulticastGroup> resource according to the Multicast Group Information and send <localMulticastGroup> create request to each Member Hosting CSE in the multicast group. See clause 7.2.2.1.

5) When the Group Hosting CSE receives the corresponding response from each Member Hosting CSE:

   a) If at least two Member Hosting CSEs respond successfully, the Group Hosting CSE shall remove the member ID(s) from the memberList for any Member Hosting CSEs whose response indicates failure.

   b) If no Member Hosting CSEs responds successfully, the Group Hosting CSE shall delete the local Multicast Group Information data object.

   c) If only one Member Hosting CSEs responds successfully, the Group Hosting CSE shall delete the local Multicast Group Information data object and send a <localMulticastGroup> delete request to that Member Hosting CSE. See clause 7.2.2.1.

7.5.3.1.3 IP multicast group creation procedure

The Group Hosting CSE shall perform the operations which are specified in clause 10.2.7.13.1 of oneM2M TS-0001 [6]:

1) Create a Multicast Group Information data object locally: set its groupID to the value of the created <group>, assign the multicastType to IP, allocate a multicastAddress, set the memberList to the members that belong to this multicast group, set responseTimeWindow according to local policy. Assign a{fanout-segment} string, and set the multicastGroupFanoutTarget to /{groupHostingCSE-ID}/ {fanout-segment}.

NOTE: For a guide to an allocation scheme of IPv4 and IPv6 multicast address spaces, reference may be made to standard documents such as IETF RFC 5771 [i.10] and IETF RFC 4291 [i.11].

2) Send a CREATE <localMulticastGroup> primitive to each Member Hosting CSE in the multicast group:

   a) Prepare the attributes of the <localMulticastGroup> resource according to the Multicast Group Information and send <localMulticastGroup> create request to each Member Hosting CSE in the multicast group. See clause 7.2.2.1.

3) When the Group Hosting CSE receives the corresponding response from each Member Hosting CSE:

   a) If at least two Member Hosting CSEs respond successfully, the Group Hosting CSE shall remove the member ID(s) from the memberList for any Member Hosting CSEs whose response indicates failure.

   b) If no Member Hosting CSE responds successfully, the Group Hosting CSE shall delete the Multicast Group Information locally.

   c) If only one Member Hosting CSEs responds successfully, the Group Hosting CSE shall delete the local Multicast Group Information data object and send a <localMulticastGroup> delete request to that Member Hosting CSE. See clause 7.2.2.1.

7.5.3.2 Multicast group Update procedure

If the *memberIDs* attribute of the <group> resource was updated, the receiver shall check the *multicastCapability* attribute of each remote Member Hosting CSE's <remoteCSE> resource according to the new *memberIDs* and then make modifications to the <localMulticastGroup> resources on the remote Member Hosting CSEs as appropriate.

The steps involved are illustrated here with an example. In this example two new members (/CSE6/aa and /CSE7/aa) are added to *memberIDs* and two of the old members (/CSE2/aa and /CSE5/aa) are deleted. Table 7.5.3.2-1 shows the old and new *memberIDs* in the example.

Table 7.5.3.2-1: memberIDs in the example <group> resource

	Member Hosting CSE1	Member Hosting CSE2	Member Hosting CSE3	Member Hosting CSE4	Member Hosting CSE5	Member Hosting CSE6	Member Hosting CSE7
multicastCapability	IP	IP	MBMS	MBMS	MBMS	MBMS	MBMS
externalGroupID			12345678901	12345678901	12345678901	12345678902	12345678902
Old memberIDs	/CSE1/aa	/CSE2/aa	/CSE3/aa	/CSE4/aa	/CSE5/aa	NA	NA
New memberIDs	/CSE1/aa	NA	/CSE3/aa /CSE3/bb	/CSE4/aa	NA	/CSE6/aa	/CSE7/aa

Prior to the update, the example group has two Multicast Group Information data objects as illustrated in Table 7.5.3.2-2.

Table 7.5.3.2-2: Old Multicast Group Information Data Objects

Group	multicastType	memberList	externalGroupID
/CSEXX/group1	IP	/CSE1/aa /CSE2/aa	
/CSEXX/group1	MBMS	/CSE3/aa /CSE4/aa /CSE5/aa	12345678901

The Group Hosting CSE shall perform the following operations:

1) If there are fewer than two new Member Hosting CSEs supporting multicast, the receiver shall delete all the local Multicast Group Information data objects and send <localMulticastGroup> delete requests to each old Member Hosting CSE before updated. See clause 7.2.2.1.

2) If there are at least two new Member Hosting CSEs supporting multicast, the receiver shall create new local Multicast Group Information data objects according to the new Member Hosting CSEs.

   In this example, the new Multicast Group Information is illustrated in Table 7.5.3.2-3. The IP multicast is deleted because there is only one Member Hosting CSE (CSE1) left. The two MBMS multicast group data objects are created because there are two different externalGroupIDs.

Table 7.5.3.2-3: New Multicast Group Information Data Object Example of <group> resource

Group	multicastType	memberList	externalGroupID
/CSEXX/group1	MBMS	/CSE6/aa /CSE7/aa	12345678902
/CSEXX/group1	MBMS	/CSE3/aa /CSE3/bb /CSE4/aa	12345678901

3) The receiver compares each new Multicast Group Information data object with the each of the old data objects:

   a) If a Member Hosting CSE exists in one of the new data objects and does not exist in any of the old ones, the receiver shall send a <localMulticastGroup> create request to that Member Hosting CSE. See clause 7.2.2.1.

   b) If a Member Hosting CSE exists in an old data object and does not exist in any of the new ones, the receiver shall send a <localMulticastGroup> delete request to that Member Hosting CSE. See clause 7.2.2.1.

   c) If a Member Hosting CSE exists in both a new data object and an old one, the receiver shall compare the memberList in the two data objects. If the result is different, the receiver shall send a <localMulticastGroup> update request to the Member Hosting CSE, see clause 7.2.2.1.

   In this example, the operations for each Member Hosting CSE are illustrated in Table 7.5.3.2-4.

Table 7.5.3.2-4: <localMulticastGroup> Operation for Member Hosting CSEs Example

Member Hosting CSE	Operation in the Request
CSE1	DELETE
CSE2	DELETE
CSE3	UPDATE
CSE4	NA
CSE5	DELETE
CSE6	CREATE
CSE7	CREATE

4) When the Group Hosting CSE receives the corresponding response from each Member Hosting CSE in the new Multicast Group Information data object, the receiver shall remove the member ID(s) from the memberList for any Member Hosting CSEs whose response indicates failure:

   a) If at least two Member Hosting CSEs exist in the data object, the receiver shall keep the Multicast Group Information data object.

   b) If no Member Hosting CSE exists in the data object, the receiver shall delete the Multicast Group Information data objects held locally.

   c) If only one Member Hosting CSE exists in the data object, the receiver shall delete the local Multicast Group Information data object and send a <localMulticastGroup> delete request to that Member Hosting CSE. See clause 7.2.2.1.

   d) The receiver shall delete all the old Multicast Group Information data objects.

   In this example, the responses for each Member Hosting CSE are illustrated in Table 7.5.3.2-5.

Table 7.5.3.2-5: <localMulticastGroup> Operation Result for Member Hosting CSEs Example

Member Hosting CSE	Operation in the Request	Result indicating in the Response
CSE1	DELETE	Failure
CSE2	DELETE	Success
CSE3	UPDATE	Failure
CSE4	NA	NA
CSE5	DELETE	Failure
CSE6	CREATE	Failure
CSE7	CREATE	Success

In this example, the new Multicast Group Information after the responses have been processed is illustrated in Table 7.5.3.2-6.

Table 7.5.3.2-6: Multicast Group Information Data Object Example of <group> resource

Group	multicastType	memberList	externalGroupID
/CSEXX/group1	MBMS	/CSE3/aa /CSE4/aa	12345678901

7.5.3.3 Multicast group Delete procedure

If there are one or more Multicast Group Information data objects associated with the deleted group, the receiver shall delete all the Multicast Group Information data objects and send <localMulticastGroup> delete requests to each Member Hosting CSE, refer to the clause 7.2.1.

7.6 Security Procedures

7.6.1 Introduction

oneM2M TS-0003 [7] specifies a range of security procedures. Clause 7.6 describes how to use the security procedures as part of the general procedures, common operations, resource type-specific procedures and primitive-specific procedures. The following security procedures are described:

- End-to-End security of primitives (ESPrim): securing a primitive so that CSEs (forwarding the primitive) do not need to be trusted with the confidentiality and integrity of the primitive.

7.6.2 Procedure for applying End-to-End Security of Primitives (ESPrim)

End-to-End Security of Primitives (ESPrim) provides an interoperable framework for securing oneM2M primitives so CSEs (forwarding the primitive) do not need to be trusted with the confidentiality and integrity of the primitive. ESPrim provides mutual authentication, confidentiality, integrity protection and a freshness guarantee (bounding the age of ESPrims). Credential management aspects and data protection aspects for ESPrim are specified in oneM2M TS-0003 [7]. Architecture-level-details for the transport of ESPrim are specified in oneM2M TS-0001 [6].

The primitive to be secured is called the inner primitive. All operations (Create, Retrieve, Update, Delete, or Notify) are allowed for an inner request primitive and inner response primitive. The inner primitives are encrypted to form an ESPrim Object. A Notify Request and corresponding Notify Response primitives (called the outer request primitive and outer response primitive) are used to transport the ESPrim Objects containing this encrypted inner request primitive and inner response primitive respectively.

There are three parallel procedures which shall be considered when using ESPrim to protect primitives:

- **Inner primitive processing:** The general procedure for the inner request primitive and corresponding inner response primitive.
- **ESPrim processing:** Encrypting inner primitives to form ESPrim objects, and decrypting ESPrim objects to obtain the inner primitives.
- **Outer primitive processing:** The general procedure for the outer request primitive and corresponding outer response primitive used to transport the ESPrim objects.

There are three actors impacted:

- Originator: The CSE or AE which originates the inner primitive procedure, the ESPrim protocol and the outer Notify primitive procedure.
- Hosting CSE: The Hosting CSE from the perspective of the outer primitive procedure, hosting the <CSEBase> or <AE> resource of the Target.
- Target CSE or AE: The CSE or AE terminating the ESPrim protocol and inner primitive procedure. The Target shall be either the Hosting CSE or an AE registered to the Hosting CSE. The Target also applies some processing of the outer primitive procedure: extracting the ESPrim objects form the outer request primitives, and composing the outer response primitives.

Other Receiver CSEs on the delivery path (apart from the Originator and Hosting CSE) process and forward the outer Notify primitives according to the general procedure in clause 7.2.2.2. These Receiver CSEs do not process the ESPrim Objects nor the inner primitives. The procedures for these Receiver CSEs are not affected when ESPrim is being used.

The present clause describes the relationship between the steps of the general procedures applied to the inner primitives, the ESPrim processing and the general procedures applied to the outer primitives.

Figure 7.6.2-1 and the following text describe the procedure for applying ESPrim to protect an exchange of inner primitives.

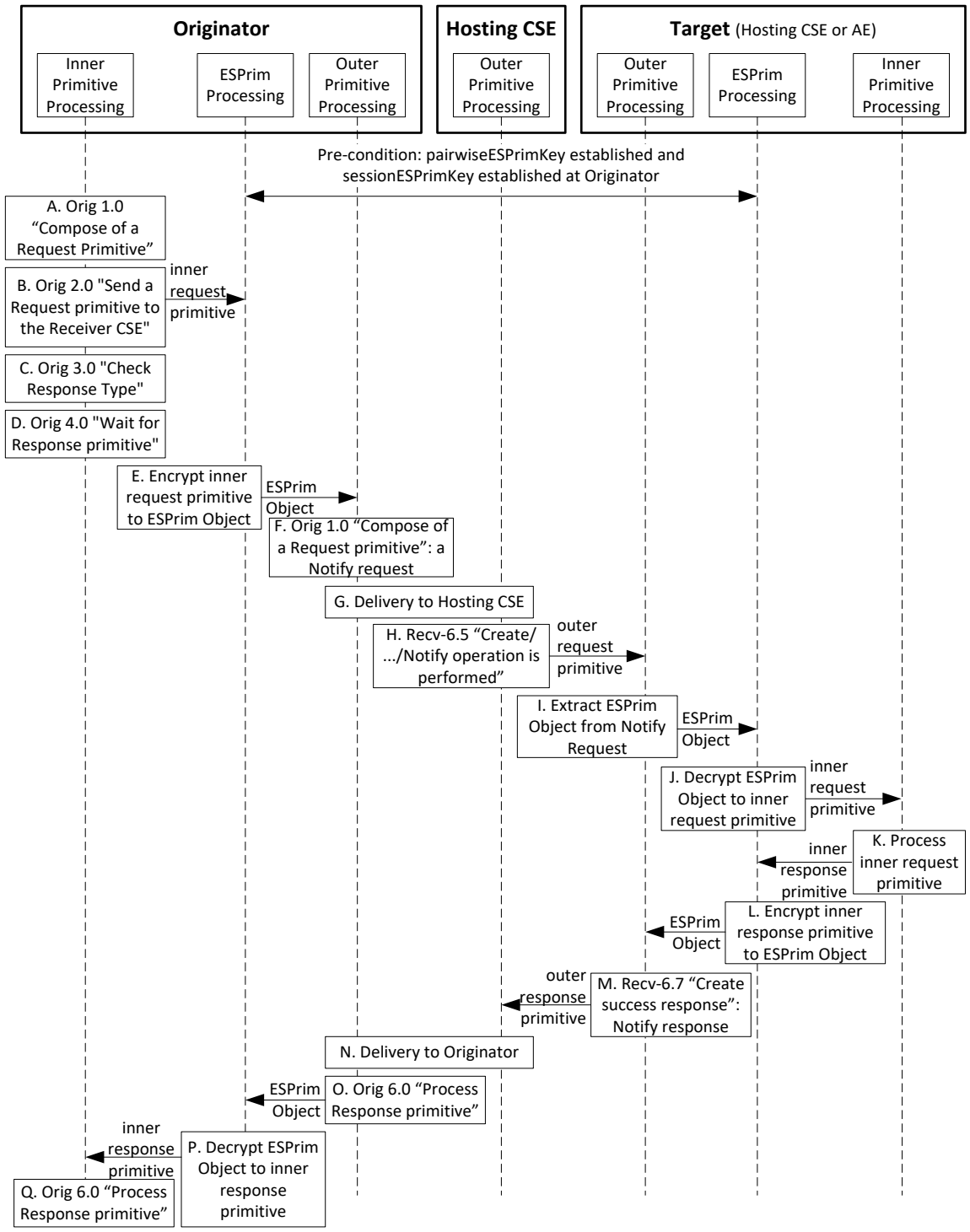

Figure 7.6.2-1: Procedure for applying End-to-End Security of Primitives (ESPrim) to protect an exchange of inner primitives

Pre-Condition: The Originator and Target have established a pairwiseESPrimKey and steps have been performed for establishing a sessionESPrimKey at the Originator (phases A and B in clause 8.4.2 of oneM2M TS-0003 [7]).

Originator:

A. The Originator's inner primitive processing shall apply Orig-1.0 "Compose of a Request primitive" to compose the inner request primitive. The inner request should not include the delivery-related parameters ***Response Type***, ***Event Category***, and ***Delivery Aggregation***.

NOTE 1: Delivery-related parameters ***Response Type***, ***Event Category***, and ***Delivery Aggregation*** in the inner request (composed at step A) will be ignored – see step K. When appropriate, these parameters are provided in the outer request primitive only (composed at step F).

B. The Originator shall not apply Orig-2.0 "Send a Request primitive to the Receiver CSE", but instead shall pass the inner request primitive to the Originator's ESPrim processing for further processing in Step E.

C. The Originator's inner primitive processing shall apply Orig-3.0 "Check Response Type". The Response Type is blockingRequest, see step A.

D. The Originator's inner primitive processing shall apply Orig-4.0 "Wait for Response primitive", entering a waiting state until the inner response primitive is received (at step P).

E. The Originator's ESPrim processing shall apply ESPrim encryption to the inner request primitive, resulting in an ESPrim Object. The ESPrim encryption process is specified in oneM2M TS-0003 [7]. The ESPrim Object is passed to the Originator's outer primitive processing. The Originator's ESPrim processing enters waiting state until the corresponding ESPrim Object is received (at step O).

F. The Originator's outer primitive processing shall apply Orig-1.0 "Compose of a Request primitive" to compose the outer request primitive. The outer request primitive shall include the following parameters:

- ***Operation***: Notify (N)
- ***To***: An address of the <CSEBase> or <AE> resource associated with the Target.
- ***From***: An address of the Originator.
- ***Request Identifier***: may be independent of the ***Request Identifier*** in the inner request primitive.
- ***Content***: a securityInfo element with child elements
  - securityInfoType: "5" (ESPrim Object).
  - esprimObject: the ESPrim Object generated at step E.

The outer request primitive may include further optional parameters as described in clause 11.4.2 of oneM2M TS-0001 [6], including the delivery-related parameters ***Response Type***, ***Event Category***, and ***Delivery Aggregation***.

G. The general procedures in clause 7.2.2.1 and clause 7.2.2.2 are followed for delivering the outer Notify request primitive from the Originator to the Hosting CSE of the addressed resource, in accordance with the communication mode of the outer request primitive. The outer Notify request primitive may be forwarded by one or more transit CSEs, which are not shown in Figure 7.6.2-1. The Originator's outer primitive processing enters a waiting state until the corresponding outer response primitive is received (at step N). The details of the delivery to and from the Hosting CSE have no impact on any other steps. The present step includes all Receiver steps in clause 7.2.2.2 up to Recv-6.4. If any errors are encountered during this step, then the Notify request primitive is rejected with a Response Status Code indicating the appropriate error code.

Hosting CSE:

H. The Hosting CSE's outer primitive processing applies Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed" to the outer Notify request primitive. This triggers "Notify processing" (clause 7.3.3.9), and the Hosting CSE passes the outer request primitive to the Target identified by the ***To*** parameter:

- If the ***To*** parameter is an address for the Hosting CSE's <CSEBase> resource, then the Target is the Hosting CSE.
- If the ***To*** parameter is an address for an <AE> resource, then the Target is the associated AE.

If any errors are encountered during this step, then the Notify request primitive is rejected with a Response Status Code indicating the appropriate error code. The Hosting CSE's outer request processing enters waiting state until the corresponding outer response primitive is received at step M.

Target:

I. The Target's outer primitive processing shall examine the securityInfoType element of the securityInfo element in the ***Content*** parameter of the outer Notify request primitive. The value "5" of the securityInfoType indicates that the securityInfo element contains an ESPrim object. The Target shall extract the ESPrim Object contained in the esprimObject element of the securityInfo element and pass the ESPrim Object to the Target's ESPrim processing.

J. The Target's ESPrim processing shall apply ESPrim decryption to the ESPrim Object, resulting in the verified inner request primitive. The ESPrim decryption process is specified in oneM2M TS-0003 [7]:

- If the Target's ESPrim processing encounters one of the error cases in Table 7.6.2-1, then the corresponding Response Status Code is returned to the Hosting CSE's outer primitive processing, and the call flow skips to step M.
- If decryption is successful, then the verified inner request primitive is passed to the Target's inner primitive processing, and the Target's ESPrim processing enters waiting state until the corresponding inner response primitive processing Object is received (at step M).

K. The Target shall process the verified inner request primitive according to the general procedure for Receiver, see clause 7.2.2.2:

- The Target shall ignore delivery-related parameters in the inner primitive, such as the ***Response Type***, ***Event Category***, or ***Delivery Aggregation***. See note 1.

An inner response primitive is returned to the Target's ESPrim processing.

NOTE 2: This call flow does not distinguish between successful and unsuccessful inner response primitives. Both will have ESPrim encryption applied prior to delivery back to the Originator.

L. The Target's ESPrim processing shall apply ESPrim encryption to the inner response primitive, resulting in an ESPrim Object. The ESPrim encryption process is specified in oneM2M TS-0003 [7]. If the Target's ESPrim processing encounters one of the error cases in Table 7.6.2-1, then the corresponding Response Status Code returned to the Hosting CSE's outer primitive processing and the call flow skips to step M. If the ESPrim processing is successful, then the ESPrim Object is passed to the Hosting Target's outer primitive processing.

M. If the Target's ESPrim processing returned a Response Status Code indicating an error, then the Hosting CSE's outer primitive processing forms an error Notify response as described in clause 7.3.3.13. If the Target's ESPrim processing returned an ESPrim Object, then the Target's outer primitive processing forms a successful Notify Response as described in clause 7.3.3.12, with the ***Content*** containing a securityInfo element with the following child elements:

- securityInfoType: "5" (ESPrim Object).
- esprimObject: the ESPrim Object received from the Target's ESPrim processing.

The outer response primitive is passed to the Hosting CSE's outer primitive processing.

Hosting CSE:

N. The general procedures in clauses 7.2.2.1 and 7.2.2.2 are followed for delivering the outer response primitive from the Hosting CSE to the Originator, in accordance with the communication mode of the outer response primitive. The outer Notify response primitive may be forwarded by one or more transit CSEs, which are not shown in Figure 7.6.2-1.

Originator:

O. The Originator's outer primitive processing shall apply Orig-6.0 "Process Response primitive" to the outer response primitive, extracting the ESPrim Object and forwarding this to the Originator's ESPrim processing.

P. The Originator's ESPrim processing shall apply ESPrim decryption to the ESPrim Object, resulting in the verified inner response primitive. The ESPrim decryption process is specified in oneM2M TS-0003 [7]. If decryption is successful, then the verified inner response primitive is passed to the Originator's inner primitive processing.

Q. The Originator's inner primitive processing shall apply Orig-6.0 "Process Response primitive" to the verified inner response primitive.

Table 7.6.2-1: End-to-End security of Primitives (ESPrim) processing error cases with corresponding error message

Error Case	Error message
The Target was unable to parse the ESPrim Object.	BAD_REQUEST
The Target does not support either the combination of protocol and AEAD algorithm used for the ESPrim Object in the Request, or the sessionESPrimKey derivation algorithm identified in the originatorESPrimRandObject.	ESPRIM_UNSUPPORTED_OPTION
The pairwiseESPrimKey Identifier in the ESPrim header is not known to the Target, or is expired.	ESPRIM_UNKNOWN_KEY_ID
The esprimRandID of the originatorESPrimRandObject in the ESPrim header is not known to the Target, or the Target decides that the originatorESPrimRandObject is expired (see note 1).	ESPRIM_UNKNOWN_ORIG_RAND_ID
The esprimRandID of the receiverESPrimRandObject in the ESPrim header is not known to the Target, or the Target decides that the receiverESPrimRandObject is expired. (see note 2).	ESPRIM_UNKNOWN_RECV_RAND_ID
The integrity verification of the ESPrim payload failed.	ESPRIM_BAD_MAC
The Target encountered some other problem while decrypting the ESPrim Object and could not obtain the inner request primitive.	ESPRIM_DECRYPTION_ERROR
The Target successfully decrypted the ESPrim Object to form the inner request primitive, but encountered a problem while encrypting the inner response primitive and could not form an ESPrim Object.	ESPRIM_ENCRYPTION_ERROR
The Target detects that the ***From*** parameter of the inner request is not the same as the ID associated with the pairwiseESPrimKey.	ESPRIM_IMPERSONATION_ERROR
NOTE 1: The Target may use the esprimRandExpiry in the originatorESPrimRandObject, or the Target's own local policy, for deciding if the originatorESPrimRandObject is expired. NOTE 2: The Target may use the esprimRandExpiry in the receiverESPrimRandObject, or the Target's own local policy, for deciding if the receiverESPrimRandObject is expired.	

8 Representation of primitives in data transfer

8.1 Introduction

This clause defines the representation of request and response primitives as XML documents, JSON texts or CBOR data format. The process of translating objects (i.e. primitives in the present context) into a format that can be stored or exchanged between network entities is commonly denoted as serialization or marshalling.

The serialization described here is used in two places:

1) It can be used when transmitting primitives over communication protocols such as HTTP, CoAP, MQTT and WebSocket. When applying a particular protocol binding, it is permitted to adapt the serialization approach, in order to make use of protocol-specific features. For example, a particular protocol binding may require that one or more primitive parameters be mapped to protocol-specific header fields rather than being included in the protocol-specific serialized JSON, XML or CBOR which represents the message body.

2) Certain instances of resource types, e.g. instances of the <delivery> resource, include serialized primitives embedded in one of their resource attributes.

In order to enable efficient communication, the short names introduced in clause 8.2 shall be applied in XML, JSON and CBOR serializations to identify primitive parameters and resource attribute names. This implies that short names are applied in any communication over the Mca, Mcc and Mcc' reference points.

8.2 Short names

8.2.1 Introduction

XML, JSON and CBOR representations require encoding of the names of primitive parameters, resource attributes, resource types and complex data type members. Whenever a protocol binding transfers such a name over a oneM2M reference point, it shall use a shortened form of that name, rather than the full name that is used elsewhere in this and other oneM2M specifications. Short names enable payload reduction on involved telecommunication interfaces.

The mapping between the full names and their shortened form is given in the clauses 8.2.2 to 8.2.5.

These names are case-sensitive. A oneM2M implementation shall use the letter casing given in these clauses.

8.2.2 Primitive parameters

In protocol bindings primitive parameter names shall be translated into short names of Table 8.2.2-1.

Table 8.2.2-1: Primitive parameter short names

Parameter Name	XSD long name	Occurs in	Short Name
Operation	operation	Request	***op***
To	to	Request, Response	***to***
From	from	Request, Response	***fr***
Request Identifier	requestIdentifier	Request, Response	***rqi***
Resource Type	resourceType	Request	***ty***
Content	primitiveContent	Request, Response	***pc***
Role IDs	roleIDs	Request	***rids***
Originating Timestamp	originatingTimestamp	Request, Response	***ot***
Request Expiration Timestamp	requestExpirationTimestamp	Request	***rqet***
Result Expiration Timestamp	resultExpirationTimestamp	Request, Response	***rset***
Operation Execution Time	operationExecutionTime	Request	***oet***
Response Type	responseType	Request	***rt***
Result Persistence	resultPersistence	Request	***rp***
Result Content	resultContent	Request	***rcn***
Event Category	eventCategory	Request, Response	***ec***
Delivery Aggregation	deliveryAggregation	Request	***da***
Group Request Identifier	groupRequestIdentifier	Request	***gid***
Filter Criteria	filterCriteria	Request	***fc***
Desired Identifier Result Type	desiredIdentifierResultType	Request	***drt***
Response Status Code	responseStatusCode	Response	***rsc***
Tokens	tokens	Request	***tkns***
Token IDs	tokenIDs	Request	***tids***
Token Request Indicator	tokenRequestIndicator	Request	***tqi***
Local Token IDs	localTokenIDs	Request	***ltids***
Group Request Target Members	groupRequestTargetMembers	Request	***grtm***
Assigned Token Identifiers	assignedTokenIdentifiers	Response	***ati***
Token Request Information	tokenRequestInformation	Response	***tqf***
Content Status	contentStatus	Response	***cnst***
Content Offset	contentOffset	Response	***cnot***
Authorization Signature Indicator	authorSignIndicator	Request	***asi***
Authorization Signature Request Information	authorSignReqInfo	Response	***asri***
Authorization Signature	authorSigns	Request	***aus***
Authorization Relationship	authorRelIndicator	Request	***auri***

Parameter Name	XSD long name	Occurs in	Short Name
Indicator			
Semantic Query Indicator	semanticQueryIndicator	Request	***sqi***
Release Version Indicator	releaseVersionIndicator	Request, Response	***rvi***
Vendor Information	vendorInformation	Request, Response	***vsi***

XML serialized representations of primitives employ root elements to differentiate between request and response primitive types (see clause 8.3). These elements are also embedded in some oneM2M-defined complex datatypes. These root element names shall be translated into short names as in Table 8.2.2-2. Their short name serialization shall include the namespace prefix.

Table 8.2.2-2: Primitive root element short names

Root Element Name	Occurs in	Short Name
m2m:requestPrimitive	Request, transaction	***rqp***
m2m:responsePrimitive	Response, aggregatedResponse, transaction	***rsp***

8.2.3 Resource attributes

In protocol bindings, resource attributes names shall be translated into short names shown in the following tables.

Table 8.2.3-1: Resource attribute short names (1/6)

Attribute Name	Occurs in	Short Name
accessControlPolicyIDs	All except accessControlPolicy, contentInstance	***acpi***
announcedAttribute	accessControlPolicy, AE, container, contentInstance, group, locationPolicy, mgmtObj, node, remoteCSE, schedule, semanticDescriptor	***aa***
announceTo	accessControlPolicy, AE, container, contentInstance, group, locationPolicy, mgmtObj, node, remoteCSE, schedule, semanticDescriptor	***at***
creationTime	All	***ct***
expirationTime	All except contentInstance, CSEBase	***et***
labels	All (optional)	***lb***|
lastModifiedTime	All	***lt***
link	All	***lnk***
parentID	All	***pi***
resourceID	All	***ri***
resourceType	All	***ty****
stateTag	container, contentInstance, delivery, request	***st***
resourceName	All	***rn***
privileges	accessControlPolicy	***pv***
selfPrivileges	accessControlPolicy	***pvs***
authorizationDecisionResourceIDs	accessControlPolicy	***adri***
authorizationPolicyResourceIDs	accessControlPolicy	***apri***
authorizationInformationResourceIDs	accessControlPolicy	***airi***
App-ID	AE	***api***
AE-ID	AE	***aei***
AE-IDList	AEContactListPerCSE	***ail***
appName	AE	***apn***
pointOfAccess	AE, CSEBase, remoteCSE	***poa***
ontologyRef	AE, container, contentInstance, semanticDescriptor. flexContainer, timeSeries	***or***
nodeLink	AE, CSEBase, remoteCSE, flexContainer	***nl***
contentSerialization	AE	***csz***
registrationStatus	AE	***regs***
trackRegistrationPoints	AE	***trps***
sessionCapabilities	AE	***scp***

Attribute Name	Occurs in	Short Name
activityPatternElements	AE, remoteCSE	***ape***
triggerEnable	AE, remoteCSE	***tren***
creator	container, contentInstance, eventConfig, group, pollingChannel, statsCollect, statsConfig, subscription, semanticDescriptor, notificationTargetPolicy, flexContainer, timeSeries, crossResourceSubscription, backgroundDataTransfer	***cr***
maxNrOfInstances	container, timeSeries	***mni***
maxByteSize	container, timeSeries	***mbs***
maxInstanceAge	container, timeSeries	***mia***
currentNrOfInstances	container, timeSeries	***cni***

Table 8.2.3-2: Resource attribute short names (2/6)

Attribute Name	Occurs in	Short Name
currentByteSize	container	***cbs***
locationID	container	***li***
disableRetrieval	container	***disr***
contentInfo	contentInstance, timeSeries	***cnf***
contentSize	contentInstance, timeSeriesInstance	***cs***
contentRef	contentInstance	***conr***
containerDefinition	flexContainer	***cnd***
primitiveContent	request	***pc****
content	contentInstance, timeSeriesInstance	***con***
cseType	CSEBase, remoteCSE	***cst***
CSE-ID	CSEBase, remoteCSE, service SubscribedNode, AEContactListPerCSE	***csi***
supportedResourceType	CSEBase	***srt***
notificationCongestionPolicy	CSEBase	***ncp***
source	delivery	***sr***
target	delivery, request	***tg***
lifespan	delivery	***ls***
eventCat	delivery	***ec***
deliveryMetaData	delivery	***dmd***
aggregatedRequest	delivery	***arq***
eventID	eventConfig, statsCollect	***evi***
eventType	eventConfig	***evt***
evenStart	eventConfig	***evs***
eventEnd	eventConfig	***eve***
operationType	eventConfig	***opt***
dataSize	eventConfig	***ds***
eventResourceTypes	eventConfig	***erts***
eventResourceIDs	eventConfig	***eris***
execStatus	execInstance	***exs***
execResult	execInstance	***exr***
execDisable	execInstance	***exd***
execTarget	execInstance, mgmtCmd	***ext***
execMode	execInstance, mgmtCmd	***exm***
execFrequency	execInstance, mgmtCmd	***exf***
execDelay	execInstance, mgmtCmd	***exy***
execNumber	execInstance, mgmtCmd	***exn***
execReqArgs	execInstance, mgmtCmd	***exra***
execEnable	mgmtCmd	***exe***
memberType	group	***mt***
specializationType	group	***spty***
currentNrOfMembers	group	***cnm***
maxNrOfMembers	group	***mnm***
memberIDs	group, backgroundDataTransfer	***mid***
membersAccessControlPolicyIDs	group	***macp***
memberTypeValidated	group	***mtv***
consistencyStrategy	group	***csy***
semanticSupportIndicator	group	***ssi***
notifyAggregation	group	***nar***
groupName	group, subscription	***gn***

Attribute Name	Occurs in	Short Name
locationSource	locationPolicy	***los***
locationUpdatePeriod	locationPolicy	***lou***
locationTargetID	locationPolicy	***lot***
locationServer	locationPolicy	***lor***
locationContainerID	locationPolicy	***loi***
locationContainerName	locationPolicy	***lon***
locationStatus	locationPolicy	***lost***
authID	locationPolicy	***aid***
retrieveLastKnownLocation	locationPolicy	***rlkl***
locationUpdateEventCriteria	locationPolicy	***luec***
locationInformationType	locationPolicy	***lit***
geographicalTargetArea	locationPolicy	***gta***
geofenceEventCriteria	locationPolicy	***gec***
description	mgmtCmd, mgmtObj, all management resources from firmware, ontology	***dc***
cmdType	mgmtCmd	***cmt***
mgmtDefinition	mgmtObj, all management resources from firmware	***mgd***
objectIDs	mgmtObj	***obis***

Table 8.2.3-3: Resource attribute short names (3/6)

Attribute Name	Occurs in	Short Name
objectPaths	mgmtObj	***obps***
mgmtSchema	mgmtObj	***mgs***
nodeID	node	***ni***
hostedCSELink	node	***hcl***
mgmtClientAddress	node	***mgca***
hostedAELinks	node	***hael***
hostedServiceLinks	node	***hsl***
networkID	node	***nid***
roamingStatus	node	***rms***
CSEBase	remoteCSE	***cb****
M2M-Ext-ID	remoteCSE, AE, locationPolicy, triggerRequest	***mei***
Trigger-Recipient-ID	remoteCSE, triggerRequest	***tri***
requestReachability	remoteCSE	***rr***
triggerReferenceNumber	remoteCSE	***trn***
descendantCSEs	remoteCSE	***dcse***
multicastCapability	remoteCSE	***mtcc***
originator	request	***org***
metaInformation	request	***mi***
requestStatus	request	***rs***
operationResult	request	***ors***
operation	request	***op****
requestID	request	***rid***
scheduleElement	schedule	***se***
networkCoordinated	schedule	***nco***
deviceIdentifier	serviceSubscribedNode	***di***
ruleLinks	serviceSubscribedNode	***rlk***
niddRequired	serviceSubscribedNode	***nrq***
statsCollectID	statsCollect	***sci***
collectingEntityID	statsCollect	***cei***
collectedEntityID	statsCollect	***cdi***
devStatus	areaNwkDeviceInfo	***ss***
statsRuleStatus	statsCollect	***srs***
statModel	statsCollect	***sm***
collectPeriod	statsCollect	***cp***
eventNotificationCriteria	subscription	***enc***
expirationCounter	subscription, crossResourceSubscription	***exc***
notificationURI	subscription, crossResourceSubscription	***nu***
groupID	subscription	***gpi***
notificationForwardingURI	subscription	***nfu***
batchNotify	subscription	***bn***
rateLimit	subscription	***rl***

Attribute Name	Occurs in	Short Name
preSubscriptionNotify	subscription	***psn***
pendingNotification	subscription	***pn***
notificationStoragePriority	subscription	***nsp***
latestNotify	subscription	***ln***
notificationContentType	subscription	***nct***
notificationEventCat	subscription, crossResourceSubscription	***nec***
subscriberURI	subscription, crossResourceSubscription	***su***
version	firmware, software, token	***vr***
URL	firmware, software	***url***
update	firmware	***ud***
updateStatus	firmware	***uds***
install	software	***in***
uninstall	software	***un***
installStatus	software	***ins***
activate	software	***act***
deactivate	software	***dea***
activeStatus	software, areaNwkInfo	***acts***
memAvailable	memory	***mma***
memTotal	memory	***mmt***

Table 8.2.3-4: Resource attribute short names (4/6)

Attribute Name	Occurs in	Short Name
areaNwkType	areaNwkInfo	***ant***
listOfDevices	areaNwkInfo	***ldv***
devId	areaNwkDeviceInfo	***dvd***
devType	areaNwkDeviceInfo	***dvt***
areaNwkId	areaNwkDeviceInfo	***awi***
sleepInterval	areaNwkDeviceInfo	***sli***
sleepDuration	areaNwkDeviceInfo	***sld***
listOfNeighbors	areaNwkDeviceInfo	***lnh***
batteryLevel	battery	***btl***
batteryStatus	battery	***bts***
deviceLabel	deviceInfo	***dlb***
manufacturer	deviceInfo	***man***
model	deviceInfo	***mod***
deviceType	deviceInfo	***dty***
fwVersion	deviceInfo	***fwv***
swVersion	deviceInfo	***swv***
hwVersion	deviceInfo	***hwv***
manufacturerDetailsLink	deviceInfo	***mfdl***
manufacturingDate	deviceInfo	***mfd***
subModel	deviceInfo	***smod***
deviceName	deviceInfo	***dvnm***
osVersion	deviceInfo	***osv***
country	deviceInfo	***cnty***
location	deviceInfo	***loc***
systemTime	deviceInfo	***syst***
supportURL	deviceInfo	***spur***
presentationURL	deviceInfo	***purl***
protocol	deviceInfo	***ptl***
capabilityName	deviceCapability	***can***
attached	deviceCapability	***att***
capabilityActionStatus	deviceCapability	***cas***
enable	deviceCapability, allJoynSvcObject	***ena***
disable	deviceCapability	***dis***
currentState	deviceCapability	***cus***
reboot	reboot	***rbo***
factoryReset	reboot	***far***
logTypeId	eventLog	***lgt***
logData	eventLog	***lgd***
logStatus	eventLog	***lgst***
logStart	eventLog	***lga***

Attribute Name	Occurs in	Short Name
logStop	eventLog	***lgo***
firmwareName	firmware	***fwn***
softwareName	software	***swn***
cmdhPolicyName	cmdhPolicy	***cpn***
mgmtLink	cmdhPolicy, activeCmdhPolicy, cmdhDefaults, cmdhNetworkAccessRules, cmdhNwAccessRule	***cmlk***
activeCmdhPolicyLink	activeCmdhPolicy	***acmlk***
order	cmdhDefEcValue, cmdhLimits	***od***
defEcValue	cmdhDefEcValue	***dev***
requestOrigin	cmdhDefEcValue, cmdhLimits	***ror***
requestContext	cmdhDefEcValue, cmdhLimits	***rct***
requestContextNotification	cmdhDefEcValue, cmdhLimits	***rctn***
requestCharacteristics	cmdhDefEcValue, cmdhLimits	***rch***
applicableEventCategories	cmdhNetworkAccessRules	***aecs***
applicableEventCategory	cmdhEcDefParamValues, cmdhBuffer	***aec***
defaultRequestExpTime	cmdhEcDefParamValues	***dqet***
defaultResultExpTime	cmdhEcDefParamValues	***dset***
defaultOpExecTime	cmdhEcDefParamValues	***doet***
defaultRespPersistence	cmdhEcDefParamValues	***drp***
defaultDelAggregation	cmdhEcDefParamValues	***dda***
limitsEventCategory	cmdhLimits	***lec***
limitsRequestExpTime	cmdhLimits	***lqet***
limitsResultExpTime	cmdhLimits	***lset***
limitsOpExecTime	cmdhLimits	***loet***
limitsRespPersistence	cmdhLimits	***lrp***
limitsDelAggregation	cmdhLimits	***lda***
targetNetwork	cmdhNwAccessRule	***ttn***

Table 8.2.3-5: Resource attribute short names (5/6)

Attribute Name	Occurs in	Short Name
minReqVolume	cmdhNwAccessRule	***mrv***
spreadingWaitTime	cmdhNwAccessRule	***swt***
backOffParameters	cmdhNwAccessRule	***bop***
otherConditions	cmdhNwAccessRule	***ohc***
maxBufferSize	cmdhBuffer	***mbfs***
storagePriority	cmdhBuffer	***sgp***
applicableCredIDs	serviceSubscribedAppRule	***apci***
allowedApp-IDs	serviceSubscribedAppRule	***aai***
allowedAEs	serviceSubscribedAppRule	***aae***
allowedRole-IDs	serviceSubscribedAppRule	***ari***
notificationTargetURI	notificationTargetMgmtPolicyRef	***ntu***
notificationPolicyID	notificationTargetMgmtPolicyRef	***npi***
action	notificationTargetPolicy	***acn***
policyLabel	notificationTargetPolicy	***plbl***
rulesRelationship	notificationTargetPolicy	***rrs***
creator	notificationTargetPolicy	***cr***
deletionRules	policyDeletionRules	***dr***
deletionRulesRelation	policyDeletionRules	***drr***
dynamicAuthorizationConsultationIDs	All resources having an accessControlPolicyID attribute	***daci***
dynamicAuthorizationEnabled	dynamicAuthorizationConsultation	***dae***
dynamicAuthorizationPoA	dynamicAuthorizationConsultation	***dap***
dynamicAuthorizationLifetime	dynamicAuthorizationConsultation	***dal***
descriptorRepresentation	semanticDescriptor	***dcrp***
semanticOpExec	semanticDescriptor	***soe***
descriptor	semanticDescriptor	***dsp***
relatedSemantics	semanticDescriptor	***rels***
semanticValidated	semanticDescriptor	***svd***
validationEnable	semanticDescriptor	***vlde***
periodicInterval	timeSeries	***pei***
missingDataDetect	timeSeries	***mdd***
missingDataMaxNr	timeSeries	***mdn***
missingDataList	timeSeries	***mdlt***

Attribute Name	Occurs in	Short Name
missingDataCurrentNr	timeSeries	***mdc***
missingDataDetectTimer	timeSeries	***mdt***
dataGenerationTime	timeSeriesInstance	***dgt***
sequenceNr	timeSeriesInstance	***snr***
roleID	role	***rlid***
roleName	role	***rlnm***
tokenLink	role	***rltl***
tokenID	token	***tkid***
tokenObject	token	***tkob***
issuer	token, role	***tkis***
holder	token, role	***tkhd***
notBefore	token, role	***tknb***
notAfter	token, role	***tkna***
tokenName	token	***tknm***
audience	token	***tkau***
permissions	token	***tkps***
extension	token	***tkex***
e2eSecInfo	CSEBase, remoteCSE, AE	***esi***
supportedReleaseVersions	CSEBase, remoteCSE, AE	***srv***

Table 8.2.3-6: Resource attribute short names (6/6)

Attribute Name	Occurs in	Short Name
serviceName	genericInterworkingService	***gisn***
operationName	genericInterworkingOperationInstance	***gion***
inputDataPointLinks	genericInterworkingService, genericInterworkingOperationInstance	***giip***
outputDataPointLinks	genericInterworkingService, genericInterworkingOperationInstance	***giop***
inputLinks	genericInterworkingOperationInstance	***giil***
outputLinks	genericInterworkingOperationInstance	***giol***
operationState	genericInterworkingOperationInstance	***gios***
direction	allJoynApp	***dir***
objectPath	allJoynSvcObject	***ajop***
interfaceIntrospectXmlRef	allJoynInterface	***ajir***
input	allJoynMethodCall	***inp***
callStatus	allJoynMethodCall	***clst***
output	allJoynMethodCall	***out***
currentValue	allJoynProperty	***crv***
requestedValue	allJoynProperty	***rqv***
decision	authorizationDecision	***dec***
status	authorizationDecision, authorizationPolicy, authorizationInformation	***sus***
to	authorizationDecision, authorizationPolicy	***to****
from	authorizationDecision, authorizationInformation	***fr****
requestedResourceType	authorizationDecision	***rrt***
operation	authorizationDecision	***op****
filterUsage	authorizationDecision	***fu***
roleIDs	authorizationDecision, authorizationInformation	***rids****
tokenIDs	authorizationDecision, authorizationInformation	***tids****
tokens	authorizationDecision	***tkns****
requestTime	authorizationDecision	***rtm***
originatorLocation	authorizationDecision	***olo***
originatorIP	authorizationDecision	***oip***
policies	authorizationPolicy	***ps***
combiningAlgorithm	authorizationPolicy	***ca***
ontologyFormat	ontology	***ontf***
ontologyContent	ontology	***ontc***
memberFilter	semanticMashupJobProfile	***mbft***
smiID	semanticMashupJobProfile	***miid***
inputDescriptor	semanticMashupJobProfile	***iptd***
outputDescriptor	semanticMashupJobProfile	***uptd***
functionDescriptor	semanticMashupJobProfile	***fucd***

Attribute Name	Occurs in	Short Name
smjpID	semanticMashupInstance	***mjid***
smjpInputParameter	semanticMashupInstance, semanticMashupResult	***jpin***
memberStoreType	semanticMashupInstance	***mst***
mashupMember	semanticMashupInstance	***msm***
resultGenType	semanticMashupInstance	***rgt***
periodForResultGen	semanticMashupInstance	***prg***
mashupResultFormat	semanticMashupResult	***mrf***
mashupResult	semanticMashupResult	***mrt***
numberImpactedCSEs	AEContactList	***nic***
externalGroupID	LocalMulticastGroup, remoteCSE	***egid***
multicastAddress	LocalMulticastGroup	***mad***
multicastGroupFanoutTarget	LocalMulticastGroup	***mgft***
memberList	LocalMulticastGroup	***mli***
responseTarget	LocalMulticastGroup	***rstt***
responseTimeWindow	LocalMulticastGroup	***rstw***
TMGI	LocalMulticastGroup	***tmgi***
sessionOriginatorID	multimediaSession	***soi***
acceptedSessionDescriptions	multimediaSession	***asd***
offeredSessionDescriptions	multimediaSession	***osd***
sessionState	multimediaSession	***sst***
triggerPurpose	triggerRequest	***tpe***
triggerStatus	triggerRequest	***tst***
triggerValidityTime	triggerRequest	***tvt***
triggerInfoAE-ID	triggerRequest	***tiae***
triggerInfoAddress	triggerRequest	***tia***
triggerInfoOperation	triggerRequest	***tio***
targetedResourceType	triggerRequest	***tirt***
triggerReference	triggerRequest	***trf***
regularResourcesAsTarget	crossResourceSubscription	***rrat***
subscriptionResourcesAsTarget	crossResourceSubscription	***srat***
timeWindowType	crossResourceSubscription	***twt***
timeWindowSize	crossResourceSubscription	***tws***
eventNotificationCriteriaSet	crossResourceSubscription	***encs***
associatedCrossResourceSub	subscription	***acrs***
volumePerNode	backgroundDataTransfer	***vpn***
numberOfNodes	backgroundDataTransfer	***non***
desiredTimeWindow	backgroundDataTransfer	***dtw***
transferSelectionGuidance	backgroundDataTransfer	***tsg***
geographicInformation	backgroundDataTransfer	***ggi***
groupLink	backgroundDataTransfer	***gli***
transactionLockTime	transactionMgmt, transaction	***tltm***
transactionExecuteTime	transactionMgmt, transaction	***text***
transactionCommitTime	transactionMgmt, transaction	***tct***
transactionExpirationTime	transactionMgmt	***tept***
transactionMode	transactionMgmt	***tmd***
transactionLockType	transactionMgmt, transaction	***tltp***
transactionControl	transactionMgmt, transaction	***tctl***
transactionState	transactionMgmt, transaction	***trst***
transactionMaxRetries	transactionMgmt	***tmr***
transactionMgmtHandling	transactionMgmt	***tmh***
requestPrimitives	transactionMgmt	***rqps***
responsePrimitives	transactionMgmt	***rsps***
transactionID	transaction	***tid***
NOTE: * marked short names have been already assigned in Table 8.2.2-1.		

8.2.4 Resource types

In protocol bindings resource type names shall be translated into short names of Table 8.2.4-1.

Table 8.2.4-1: Resource and specialization type short names

Resource Type Name	Short Name
accessControlPolicy	***acp***
accessControlPolicyAnnc	***acpA***
AE	***ae***
AEAnnc	***aeA***
container	***cnt***
containerAnnc	***cntA***
contentInstance	***cin***
contentInstanceAnnc	***cinA***
CSEBase	***cb***
delivery	***dlv***
eventConfig	***evcg***
execInstance	***exin***
group	***grp***
groupAnnc	***grpA***
locationPolicy	***lcp***
locationPolicyAnnc	***lcpA***
m2mServiceSubscriptionProfile	***mssp***
mgmtCmd	***mgc***
node	***nod***
nodeAnnc	***nodA***
pollingChannel	***pch***
remoteCSE	***csr***
remoteCSEAnnc	***csrA***
request	***req***
schedule	***sch***
scheduleAnnc	***schA***
serviceSubscribedAppRule	***asar***
serviceSubscribedNode	***svsn***
statsCollect	***stcl***
statsConfig	***stcg***
subscription	***sub***
firmware	***fwr***
firmwareAnnc	***fwrA***
software	***swr***
softwareAnnc	***swrA***
memory	***mem***
memoryAnnc	***memA***
areaNwkInfo	***ani***
areaNwkInfoAnnc	***aniA***
areaNwkDeviceInfo	***andi***
areaNwkDeviceInfoAnnc	***andiA***
battery	***bat***
batteryAnnc	***batA***
deviceInfo	***dvi***
deviceInfoAnnc	***dviA***
deviceCapability	***dvc***
deviceCapabilityAnnc	***dvcA***
reboot	***rbo ****
rebootAnnc	***rboA***
eventLog	***evl***
eventLogAnnc	***evlA***
cmdhPolicy	***cmp***
activeCmdhPolicy	***acmp***
cmdhDefaults	***cmdf***
cmdhDefEcValue	***cmdv***
cmdhEcDefParamValues	***cmpv***
cmdhLimits	***cml***
cmdhNetworkAccessRules	***cmnr***
cmdhNwAccessRule	***cmwr***
cmdhBuffer	***cmbf***
notificationTargetMgmtPolicyRef	***ntpr***
notificationTargetPolicy	***ntp***

Resource Type Name	Short Name
policyDeletionRules	***pdr***
dynamicAuthorizationConsultation	***dac***
semanticDescriptor	***smd***
semanticDescriptorAnnc	***smdA***
timeSeries	***ts***
timeSeriesAnnc	***tsA***
timeSeriesInstance	***tsi***
timeSeriesInstanceAnnc	***tsiA***
role	***rol***
token	***tk***
genericInterworkingService	***gis***
genericInterworkingService*Annc*	***gisA***
genericInterworkingOperationInstance	***gio***
genericInterworkingOperationInstance*Annc*	***gioA***
svcObjWrapper	***ajsw***
svcObjWrapper*Annc*	***ajswA***
svcFwWrapper	***ajfw***
svcFwWrapper*Annc*	***ajfwA***
allJoynApp	***ajap***
allJoynApp*Annc*	***ajapA***
allJoynSvcObject	***ajso***
allJoynSvcObject*Annc*	***ajsoA***
allJoynInterface	***ajif***
allJoynInterface*Annc*	***ajifA***
allJoynMethod	***ajmd***
allJoynMethod*Annc*	***ajmdA***
allJoynMethodCall	***ajmc***
allJoynMethodCall*Annc*	***ajmcA***
allJoynProperty	***ajpr***
allJoynProperty*Annc*	***ajprA***
authorizationDecision	***auds***
authorizationPolicy	***aupy***
authorizationInformation	***auif***
ontologyRepository	***ontr***
ontologyRepositoryAnnc	***ontrA***
ontology	***ont***
ontologyAnnc	***ontA***
semanticMashupJobProfile	***smjp***
semanticMashupJobProfile*Annc*	***smjpA***
semanticMashupInstance	***smi***
semanticMashupInstance*Annc*	***smiA***
semanticMashupResult	***smr***
semanticMashupResult*Annc*	***smrA***
AEContactList	***alst***
AEContactListPerCSE	***alpc***
localMulticastGroup	***lmg***
multimediaSession	***mms***
multimediaSessionAnnc	***mmsA***
triggerRequest	***tgr***
crossResourceSubscription	***crs***
backgroundDataTransfer	***bdt***
transaction	***trac***
transactionMgmt	***tram***
NOTE: * marked short names have been already assigned in attribute Tables 8.2.3-1 to 8.2.3-5.	

8.2.5 Complex data types members

In protocol bindings complex data types member names shall be translated into short names of Table 8.2.5-1.

Table 8.2.5-1: Complex data type member short names

Member Name	Occurs in	Short Name
createdBefore	filterCriteria, eventNotificationCriteria	***crb***
createdAfter	filterCriteria, eventNotificationCriteria	***cra***
modifiedSince	filterCriteria, eventNotificationCriteria	***ms***
unmodifiedSince	filterCriteria, eventNotificationCriteria	***us***
stateTagSmaller	filterCriteria, eventNotificationCriteria	***sts***
stateTagBigger	filterCriteria, eventNotificationCriteria	***stb***
expireBefore	filterCriteria, eventNotificationCriteria	***exb***
expireAfter	filterCriteria, eventNotificationCriteria	***exa***
labels	filterCriteria, eventNotificationCriteria	***lbl****
labelsQuery	filterCriteria	***lbq***
resourceType	filterCriteria, accessControlObjectDetails	***ty****
sizeAbove	filterCriteria, eventNotificationCriteria	***sza***
sizeBelow	filterCriteria, eventNotificationCriteria	***szb***
contentType	filterCriteria	***cty***
limit	filterCriteria	***lim***
attribute	filterCriteria, eventNotificationCriteria	***atr***
contentFilterSyntax	filterCriteria	***cfs***
contentFilterQuery	filterCriteria	***cfq***
level	filterCriteria	***lvl***
offset	filterCriteria	***ofst***
notificationEventType	eventNotificationCriteria, notificationEvent	***net***
operationMonitor	eventNotificationCriteria, notificationEvent	***om***
representation	notificationEvent	***rep***
filterUsage	filterCriteria	***fu****
eventCatType	eventCat	***ect***
eventCatNo	eventCat	***ecn***
number	batchNotify	***num***
duration	batchNotify	***dur***
notification	aggregatedNotification, Request Primitive Content	***sgn***
notificationEvent	notification	***nev***
verificationRequest	notification	***vrq***
subscriptionDeletion	notification	***sud***
subscriptionReference	notification	***sur***
creator	notification	***cr****
notificationForwardingURI	notification	***nfu****
notificationTarget	notification	***ntt***
targetRemovalRequest	notification	***trr***
targetRemovalAllowance	notification	***tra***
IPEDiscoveryRequest	notification	***idr***
AERegistrationPointChange	notification	***aerp***
AEReferenceIDChange	notification	***aerid***
trackingID1	notification	***tid1***
trackingID2	notification	***tid2***
filterCriteria	IPEDiscoveryRequest	***fc****
operation	operationMonitor, dynAuthDasRequest	***op****
operations	operationMonitor	***ops***
originator	operationMonitor, IPEDiscoveryRequest, dynAuthDasRequest	***or****
action	actionStatus	***acn****
status	actionStatus	***sus****
childResource	All except execInstance, announced resource, management resources from firmware	***ch***
accessControlRule	privileges, selfPrivileges	***acr***
accessControlOriginators	accessControlRule	***acor***
accessControlOperations	accessControlRule	***acop***
accessControlContexts	accessControlRule	***acco***

Member Name	Occurs in	Short Name
accessContro Window	accessControlContexts	***actw***
accessControlIpAddresses	accessControlContexts	***acip***
ipv4Addresses	accessControlIpAddress	***ipv4***
ipv6Addresses	accessControlIpAddress	***ipv6***
accessControlLocationRegion	accessControlContexts	***aclr***
countryCode	accessControlLocationRegion	***accc***
circRegion	accessControlLocationRegion	***accr***
name	attribute, anyArgType, mgmtLinkRef, childResourceRef, contentRef	***nm****
specializationType	childResourceRef, accessControlObjectDetails	***spty****
containerDefinition	specializationType	***cnd****
mgmtDefinition	specializationType	***mgd****
value	attribute	***val***
type	anyArgType, childResourceRef, mgmtLinkRef	***typ***
maxNrOfNotify	rateLimit	***mnn***
timeWindow	rateLimit	***tww***
scheduleEntry	scheduleElement	***sce***
aggregatedNotification	Request Primitive Content	***agn***
attributeList	Request Primitive Content	***atrl***
securityInfo	Request Primitive Content, Response Primitive Content	***seci***
aggregatedResponse	Response Primitive Content	***agr***
resource	Response Primitive Content	***rce***
URIList	Response Primitive Content	***uril***
debugInfo	Response Primitive Content	***dbg***
queryResult	Response Primitive Content	***qres***
anyArg	resetArgsType, rebootArgsType, uploadArgsType, downloadArgsType, softwareInstallArgsType softwareUpdateArgsType, softwareUninstallArgsType, execReqArgsListType	***any***
fileType	downloadArgsType	***ftyp***
URI	resourceWrapper, dynAuthTokenReqInfo	***uri***
URL	downloadArgsType	***url****
username	uploadArgsType, downloadArgsType, softwareUpdateArgsType, softwareUninstallArgsType	***unm***
password	uploadArgsType, downloadArgsType, softwareUpdateArgsType, softwareUninstallArgsType	***pwd***
filesize	downloadArgsType	***fsi***
targetFile	downloadArgsType	***tgf***
delaySeconds	downloadArgsType	***dss***
successURL	downloadArgsType	***surl***
startTime	downloadArgsType	***stt***
completeTime	downloadArgsType	***cpt***
UUID	softwareInstallArgsType softwareUpdateArgsType, softwareUninstallArgsType	***uuid***
executionEnvRef	softwareInstallArgsType softwareUpdateArgsType, softwareUninstallArgsType	***eer***
version	softwareUninstallArgsType, tokenClaimSet	***vr****
reset	execReqArgsListType	***rst***
reboot	execReqArgsListType	***rbo****
upload	execReqArgsListType	***uld***
download	execReqArgsListType	***dld***
softwareInstall	execReqArgsListType	***swin***
softwareUpdate	execReqArgsListType	***swup***
softwareUninstall	execReqArgsListType	***swun***
tracingOption	deliveryMetaData	***tcop***
tracingInfo	deliveryMetaData	***tcin***

Member Name	Occurs in	Short Name
responseTypeValue	responseTypeInfo	***rtv***
notificationURI	responseTypeInfo	***nu***
timeOfDay	deletionContexts	***tod***
locationRegions	deletionContexts	***lr***
URIReference	contentRef	***urir***
semanticsFilter	filterCriteria	***smf***
missingDataList	timeSeries	***mdl***
missingData	eventNotificationCriteria	***md***
tokenID	tokenClaimSet, dynAuthLocalTokenIdAssignments	***tkid***
holder	tokenClaimSet	***tkhd****
issuer	tokenClaimSet	***tkis****
notBefore	tokenClaimSet	***tknb****
notAfter	tokenClaimSet	***tkna****
tokenName	tokenClaimSet	***tknm****
audience	tokenClaimSet	***tkau****
permissions	tokenClaimSet	***tkps****
extension	tokenClaimSet	***tkex****
permission	tokenPermissions	***pm***
resourceIDs	tokenPermission	***ris***
privileges	tokenPermission, setOfPermissions	***pv****
roleIDs	tokenPermission	***rids****
localTokenIdAssignment	dynAuthLocalTokenIdAssignments	***ltia***
localTokenID	dynAuthLocalTokenIdAssignment	***lti***
dasInfo	dynAuthTokenReqInfo	***dasi***
dasRequest	dynAuthTokenReqInfo	***daq***
securedDasRequest	dynAuthTokenReqInfo	***sdr***
filterOperation	filterCriteria, eventNotificationCriteria	***fo***
targetedResourceType	dynAuthDasRequest	***trt***
originatorIP	dynAuthDasRequest	***oip****
ipv4Address	dynAuthDasRequest, ipAddress	***ip4***
ipv6Address	dynAuthDasRequest, ipAddress	***ip6***
originatorLocation	dynAuthDasRequest	***olo****
originatorRoleIDs	dynAuthDasRequest	***orid***
requestTimestamp	dynAuthDasRequest	***rts***
targetedResourceID	dynAuthDasRequest	***trid***
proposedPrivilegesLifetime	dynAuthDasRequest	***ppl***
roleIDsFromACPs	dynAuthDasRequest	***rfa***
tokenIDs	dynAuthDasRequest	***tids***
dynamicACPInfo	dynAuthDasResponse	***dai***
grantedPrivileges	dynAuthDasResponse	***gp***
privilegesLifetime	dynAuthDasResponse	***pl***
tokens	dynAuthDasResponse	***tkns***
securityInfoType	securityInfo	***sit***
dasRequest	securityInfo	***dreq***
dasResponse	securityInfo	***dres***
dynAuthRelMapRequest	securityInfo	***darq***
dynAuthRelMapResponse	securityInfo	***dars***
esprimRandObject	securityInfo	***ero***
esprimObject	securityInfo	***epo***
escertkeMessage	securityInfo	***eckm***
resourceRef	listOfChildResourceRef	***rrf***
resourceRefList	Response Primitive Content	***rrl***
esprimRandID	originatorESPrimRandObject, receiverESPrimRandObject	***esri***
esprimRandValue	originatorESPrimRandObject, receiverESPrimRandObject	***esrv***
esprimRandExpiry	originatorESPrimRandObject, receiverESPrimRandObject	***esrx***
esprimKeyGenAlgID	originatorESPrimRandObject	***esk***
esprimKeyGenAlgIDs	receiverESPrimRandObject	***esks***
esprimProtocolAndAlgIDs	originatorESPrimRandObject, receiverESPrimRandObject	***espa***
supportede2ESecFeatures	e2eSecInfo	***esf***

Member Name	Occurs in	Short Name
certificates	e2eSecInfo	***escert***
sharedReceiverESPrimRandObject	e2eSecInfo	***esro***
networkAction	backOffParameters	***nwa***
initialBackoffTime	backOffParameters	***ibt***
additionalBackoffTime	backOffParameters	***abt***
maximumBackoffTime	backOffParameters	***mbt***
optionalRandomBackoffTime	backOffParameters	***rbt***
backOffParametersSet	backOffParameters	***bops***
dataLink	listOfDataLinks	***dali***
attributeName	dataLink	***atn***
dataContainerID	dataLink	***dcid***
accessControlAuthenticationFlag	accessControlRule	***acaf***
accessControlObjectDetails	accessControlRule	***acod***
dataLinkEntry	listOfDataLinks	***dle***
childResourceType	accessControlObjectDetails, eventNotificationCriteria, filterCriteria	***chty***
parentResourceType	filterCriteria	***pty***
childLabels	filterCriteria	***clbl***
parentLabels	filterCriteria	***palb***
childAttribute	filterCriteria	***catr***
parentAttribute	filterCriteria	***patr***
applyRelativePath	filterCriteria	***arp***
sessionDescription	sessionDescriptions	***sdc***
activityPattern	activityPatternElements	***apt***
stationaryIndication	activityPattern	***sti***
dataSizeIndicator	activityPattern	***dsi***
eventNotificationCriteriaEntry	eventNotificationCriteriaSet	***encn***
memberURI	mashupMembers	***muri***
memberValue	mashupMembers	***mvl***
NOTE: * marked short names have been already assigned in attribute Table 8.2.3-1 to Table 8.2.3-6.		

8.2.6 Trigger payload fields

Trigger payload fields shall be translated into short names of Table 8.2.6-1.

Table 8.2.6-1: Trigger payload field short names

Member Name	Short Name
triggerPayload (root element)	***tgp***
triggerPurpose	***tpe****
triggerInfoAddress	***tia****
triggerInfoOperation	***tio****
triggerInfoResourceType	***tirt****
triggerInfoAE-ID	***tiae****
triggerInfoPoA	***tipa***
triggerInfoSerializationTypes	***tist***
NOTE: * marked short names have been already assigned in attribute Table 8.2.3-1 to Table 8.2.3-6.	

8.3 XML serialization

8.3.1 Method

XML serialization of request or response primitives refers to the process of representing the primitive as an XML document.

The XML document shall be a well-formed XML document compliant with W3C XML 1.0 [1]. It shall be restricted to Unicode characters and encoded using UTF-8 as described in IETF RFC 3629 [21].

The structure and data types of XML serialized request and response primitives shall be consistent with the XSD defined in CDT-requestPrimitive-v3_11_0.xsd and CDT-responsePrimitive-v3_11_0.xsd, respectively. The data types used in these XSD files comply with the definitions in clause 6 and clause 7 of the present document.

XML serializations shall comply with the order of resource attributes and elements imposed by the XML schema definition. If an implementation uses modified XSD modified from the original files for schema validation of partial resource representations (see note 2 in clause 6.1), the order of resource attributes shall not be changed.

If an *element* instance is NULL then it is serialized into the XML as an *empty element* (as defined in W3C XML 1.0 [1]) regardless of the data type that it has in the corresponding XSD.

Note that the XSD files included in the present release employ the long names for primitive parameters and other XML elements and attributes, but the primitive serialization is required to use the corresponding short names (as defined clause 8.2 of the present document).

NOTE: XML Schema files are available with both long and short names.

The primitive ***Content*** parameter is serialized just like any other element of complex type. Generally, the ***Content*** parameter may include only a partial set of attributes specified for the resource type as indicated in the ***Resource Type*** parameter, e.g. for partial Update or Retrieve Request procedures. For Notification Request primitives, the ***Content*** parameter includes a Notification data object as defined in clause 7.5.1.1 and the datatype definition given in CDT-notification-v3_11_0.xsd.

8.3.2 Examples

An example that shows a request primitive serialized into an XML document is shown below. This example shows the create request for an instance of a <contentInstance> resource. Only mandatory primitive parameters and resource attributes are shown.

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:rqp xmlns:m2m="http://www.onem2m.org/xml/protocols">
    <op>1</op>
    <to>//example.net/myCSE/-/Cont1</to>
    <fr>/myCSE/C2345</fr>
    <rqi>0002bf63</rqi>
    <ty>4</ty>
    <pc>
        <m2m:cin>
            <cnf>application/xml:1</cnf>
            <con>PHRpbWU+MTc4ODkzMDk8L3RpbWU+PHRlbXA+MjA8L3RlbXA+DQo=</con>
        </m2m:cin>
    </pc>
</m2m:rqp>
```

The XML elements have the following meaning:

- rqp: Root element of the Request primitive, which includes a reference to an XSD file which defines its datatype.
- op: ***Operation*** parameter of datatype m2m:operation: in this example value = 1 indicates a "Create" operation.
- to: ***To*** parameter of type xs:anyURI: URI of the target resource.
- fr: ***From*** parameter of type m2m:ID: ID of the Originator (either AE-ID or CSE-ID).
- rqi: ***Request Identifier*** parameter of type m2m:requestID: this could e.g. represent a counter number.
- ty: ***Resource Type*** parameter of datatype m2m:resourceType: indicating type of the resource to be created (value = 4 indicates that a <contentInstance> resource shall be created).
- pc: ***Content*** parameter of datatype m2m:primitiveContent: the attributes of the resource to be provided by the Originator.

- cin: Root element of the <contentInstance> resource of datatype m2m:contentInstance: this includes the mandatory attributes (and optional attributes not shown in this example) supplied by the request Originator. In this example, the ***Content*** parameter includes an instance of a <contentInstance> resource which consists of two attributes: *contentInfo* (cnf) – which specifies base64 encoding - and the *content* (con) itself.

The following example shows the ***Content*** of a request deleting the *maxNumberOfMembers* attribute of a <group> resource. This is done by sending a NULL value which is represented in XML as an *empty element.*

```
<?xml version="1.0" encoding="UTF-8"?>
    <m2m:grp>
        <mnm></mnm>
    </m2m:grp>
```

8.4 JSON serialization

8.4.1 Terminology

The following conventions are used in clause 8.4.2.

- The italicized terms *object*, *member*, *name*, *array*, *number*, *string*, *boolean* and *null* are to be interpreted as in IETF RFC 8259 [19].
- The italicized term *element* is to be interpreted to encompass oneM2M Primitive Parameters, Resource Attributes and other elements or attributes used inside oneM2M complex type definitions.

8.4.2 Method

The primitive shall be encoded as a JSON *object*, conforming to the requirements of IETF RFC 8259 [19]. This JSON *object* shall be restricted to Unicode characters defined in The Unicode Standard and encoded using UTF-8 as described in IETF RFC 3629 [21]. The names in each *object* in the JSON shall be unique.

The structure of the top-level primitive *object* shall be determined by the data type definitions in clause 6 and clause 7 of the present document, as follows:

1) All *member*'s *names* shall use the short name defined in clause 8.2.
2) If an *element* is defined in the present document as having a complex type, then it is serialized in the JSON *member* as an *object* and its children are recursively serialized as members of that *object*, using short names as defined in clause 8.2.
3) Where an *element* has a Global Element Declaration in the XSD its member name in the JSON serialization shall be prefixed with a namespace identifier followed by a ":" character. In particular, if the member serializes a Resource defined in the present specification its name shall have the prefix "m2m:".
4) The membership of each nested *object* shall respect the cardinality constraints from the corresponding XSD complex type definition.
5) If an *element* is defined in the present document as having an atomic data type that is numeric (including enumeration data types in clause 6.3.4) then its value is serialized into the JSON *member* as a *number*.
6) If an *element* is defined as having an atomic data type that is non-numeric then its value is serialized into the JSON *member* as a *string*.
7) If an *element* is defined as xs:boolean (or a type derived from xs:boolean) then it is serialized in the JSON *member* as a *boolean*.
8) If an *element* is defined as having an xs:list type in the corresponding XSD then it is serialized in the JSON *member* as an *array*. A list *element* that contains no values shall be represented as an empty array.
9) If an *element* instance has a NULL value then it is serialized into the JSON *member* as a *null*, regardless of the data type that it has in the corresponding XSD.

10) If an *element* is defined as having maxOccurs > 1 in the corresponding XSD then its occurrences are serialized in a single JSON *member* as an *array*.

11) If an *element* has an XSD data type that is a simple type with XML attributes, then it is serialized in the JSON member as an *object*. The XML attributes appear as *members* of that object (using their short names) and the value of the *element* is serialized as a *member* of that *object* with the special short *name* "val" (lower case).

12) The *members* (at each level) may be serialized in any order. The order in which they appear in the corresponding XSD file is immaterial.

13) If an *element* has an XSD data type that is a complex type with XML attributes, then it is serialized in JSON as an *object*. The XML attributes appear as *members* of that object (using their short names) as do the XML elements.

The ***Content*** parameter is serialized as an object containing a single member, as defined in the first column of Tables 7.5.2-1 and 7.5.2-2 using the short name from Table 8.2.4-1 or Table 8.2.3-2.

JSON serialized representations of request and response primitives shall not be encapsulated under member names m2m:rqp and m2m:rsp. Note that this is in contrast to XML serialized representations of primitives which shall include such root elements in order to assert XSD compliance, see clause 8.3.2.

8.4.3 Examples

EXAMPLE 1 starts:

An example of a request message serialized using JSON is given below:

```
{
    "op": 1,
    "fr": "Clight_ae1",
    "to": "/homegateway/light",
    "rqi": "A1234",
    "pc": {
        "m2m:sch": {
            "se": {
                "sce": ["* 0-5 2,6,10 * * * *"]
            }
        }
    },
    "ty": 18
}
```

- op: operation (in this case it is Create)
- fr: ID of the Originator (an AE in this example)
- to: URI of the target resource
- rqi: request identifier (this is a string)
- pc: attributes of the <schedule> resource, with member name "`m2m:sch`", as provided by the Originator. This is serialized as a nested JSON object
- ty: type of resource to be created (in this case a Schedule resource). This is a number.

Note that the ***Operation*** (***op***) parameter is present only in Request primitives. The presence of this parameter in JSON serialized primitive representations allows to differentiate Request primitives from Response primitives.

EXAMPLE 1 ends.

EXAMPLE 2 starts:

The next example shows an <AE> resource serialized using JSON.

The top level member, `m2m:ae`, is an object whose name consists of the prefix `m2m:` followed by the short name for the <AE> resource defined in Table 8.2.4-1. The members of this object are the attributes of <AE> using the short names from Table 8.2.3-2.

In this example the <AE> has just two direct child resources.They are <container> resources (resource type 3) and the second of them has a child <subscription> (resource type 23). The ***Result Content*** parameter in the request that was used to retrieve the <AE> resource was set to “attributes and child resource references” and the ***Filter Criteria*** *level* was 2.

The `ch` member is an array containing references to the child resources. Note the use of the special short name `val` to hold the reference itself, as specified by clause 8.4.2, rule 10.

```
{
 "m2m:ae": {
   "rn": "appname",
   "aei": "CAE01",
   "ct": "20160404T132648",
   "et": "20160408T004648",
   "lt": "20160404T132648",
   "pi": "ONET-CSE-02",
   "ri": "REQID1",
   "ty": 2,
   "ch": [{"nm":"container1", "typ":3,  "val":"mn-cse/appname/container1"},
          {"nm":"container2", "typ":3,  "val":"mn-cse/appname/container2"},
          {"nm":"sub1", "typ":23, "val":"mn-cse/appname/container2/sub1"}]
 }
}
```

EXAMPLE 2 ends.

EXAMPLE 3 starts:

The third example shows the same <AE> resource, but this time shows what would have been returned if the ***Result Content*** parameter in the request had been set to "attributes and child resources".

The child resources are serialized inline, using the shortnames from short names from Table 8.2.4-1. The child resources are nested according to their parent-child relationship, so the subscription resource (sub1) appears nested inside its parent (container2).

Since the inlined resources have XML Schema global element definitions they appear in the JSON with the prefix `m2m:` as required by clause 8.4.2, rule 3. As there can be more than one occurrence of each child resource type, they are serialized as JSON arrays as required by clause 8.4.2, rule 11.

For brevity this example does not show all the attributes of the child resources.

```
{
 "m2m:ae": {
   "rn": "appname",
   "aei": "CAE01",
   "ct": "20160404T132648",
   "et": "20160408T004648",
   "lt": "20160404T132648",
   "pi": "ONET-CSE-02",
   "ri": "REQID1",
   "ty": 2,
   "m2m:cnt":[{"rn":"container1", "ty":3,  …},
              {"rn":"container2", "ty":3,  … ,
               "m2m:sub":[{"rn":"sub1", "ty":23, …}]}
             ]
  }
}
```

EXAMPLE 3 ends.

8.5 CBOR serialization

8.5.1 Method

Concise Binary Object Representation (CBOR) is a binary serialization format of structured data specified in IETF RFC 7049 [39]. CBOR provides unambiguous encoding of structured data into a binary representation and reverse decoding.

The specifics on how CBOR can be negotiated between protocol endpoints is protocol specific and defined by the individual bindings.

This clause defines the relationship between JSON objects as defined in clause 8.4 and CBOR representations.

Section 2 of IETF RFC 7049 [39] specifies the applicable CBOR encoding rules.

In particular, the following rules shall apply when using CBOR serialization:

- Text strings (i.e. any names/keys and text string values) shall be encoded as UTF-8 strings, CBOR major type 3.
- Integer numbers shall be encoded as CBOR major types 0 or 1.
- Floating point numbers shall be encoded as CBOR major type 7 with Additional Information 26 for single precision (32-bit) and Additional Information 27 for double precision (64-bit) formats.

Note that CBOR ignores whitespace characters (including space, LF/CR) if used for formatting of JSON objects in textual representations.

If decoding of CBOR serializations results in JSON objects with member names or values not compliant with the underlying XSD, this shall be interpreted as an error by the receiver of the primitive.

8.5.2 Examples

This clause presents some examples of CBOR serialized primitives. Note that due to given encoding options, a CBOR encoder may produce somewhat different binary serializations. However, in any case the CBOR decoding shall produce an equivalent representation in JSON format as shown in the examples below.

EXAMPLE 1:

JSON representation (a request primitive of message length: 160 bytes):

```
{"op":1,"to":"//example.net/mncse1234","rqi":"A1000",
"rcn":7,"pc":{"m2m:ae":{"rn":"SmartHomeApplication", "api":"Na56", "apn":"app1234"}},"ty":2}
```

CBOR representation as sequence of hexadecimal characters (length: 108 bytes):

a6427063a1466d326d3a6165a342726e54536d617274486f6d654170706c69636174696f6e43617069444e61353643617
06e476170703132333442747902427 46f572f2f6578616d706c652e6e65742f6d6e6373653132333443726 36e07426f700
1437271694541313030 30

EXAMPLE 2:

JSON representation (a response primitive of message length: 266 bytes):

```
{"rsc":2001,"rqi":"A1000","pc":{"m2m:ae":{"rn":"SmartHomeApplication","ty":2,"ri":"ae1","api":"Na56"
,"apn":"app1234","pi":"cb1","ct":"20160506T153208",
"lt":"20160506T153208","acpi":["acp1","acp2"],"et":"20180506T153208", "aei":"S_SAH25"}}}
```

CBOR representation as sequence of hexadecimal characters (length: 178 bytes):

a3427063a1466d326d3a6165ab43617069444e6135364361706e476170703132333442657 44f323031383035303654313
533323038426374 4f32303136303530365431353332303842747902427269461653142 6c744f323031363035303654313
533323038436165694753 5f534148323542726e54536d617274486f6d654170706c69636174696f6e4270694363623144
616370698244616370314461637032437271694541313030304372736319 07d1

EXAMPLE 3:

JSON representation (request primitive of message length: 174 bytes):

```
{"op":1,"to":"//example.net/mncse1234/SmartHomeApplication",
"rqi":"A1001","rcn":7,"pc":{"m2m:cnt":{"rn":"SmartHomeContainer","mbs":100000, "mni":500}},"ty":3}
```

CBOR representation as sequence of hexadecimal characters (length: 124 bytes):

a6427063a1476d326d3a636e74a3436d6e691901f442726e52536d617274486f6d65436f6e7461696e6572436d62731a00
0186a04274790342746f582c2f2f6578616d706c652e6e65742f6d6e637365313233342f536d617274486f6d654170706c6
9636174696f6e4372636e07426f700143727169454131303031

EXAMPLE 4:

JSON representation (response primitive of message length: 393 bytes):

```
{"rsc":2001,"rqi":"A1001","pc":{"m2m:cnt":{"rn":"SmartHomeContainer",
"ty":3,"ri":"cnt1","pi":"ae1","ct":"20160506T154048",
"lt":"20160506T154048","acpi":["acp1"],"et":"20180506T154048","cr":"
S_SAH25","st":0,"mni":500,"mbs":100000,"cni":0,"cbs":0,"mia":3600}}}
```

CBOR representation as sequence of hexadecimal characters (length: 188 bytes):

a3427063a1476d326d3a636e74af4363627300427269446 36e7431436d6e691901f442637247535f53414832354265744f
323031383035303654313534303438426 3744f32303136303530365431353430343843 6d62731a000186a04274790343
6d6961190e1042737400426c744f32303136303530365431353430343842726e52536d617274486f6d65436f6e7461696
e6572427069436165314461637069814461637031436 36e690043727169454131303031437273631907d1

9 Mcn procedure

9.1 Introduction

The following clauses describe procedural details and message format bindings for various Mcn procedures.

9.2 Triggering

9.2.1 Introduction

A trigger originator (i.e. IN-CSE) may send a trigger request to an underlying network that addresses a trigger recipient (i.e. ASN/MN-CSE or an ADN-AE). A trigger request may include a payload. If the trigger has no payload, the trigger recipient shall just re-establish a network connection, so that the trigger originator can send requests to the trigger recipient. If the request contains a payload, the trigger recipient shall re-establish the network connection and perform additional actions as requested by the payload. The trigger payload fields are described in Table 9.2.1-1.

Table 9.2.1-1: Trigger payload short names and field descriptions

Field Name	Request Optionality			Data Type	Default Value and Constraints
	Establish Connection	registration Request	execute CRUD		
triggerPurpose	M	M	M	m2m:triggerPurpose	If a trigger has a payload then this field is mandatory and shall be specified by the trigger originator. If a trigger does not have a payload then the default *triggerPurpose* is establishConnection.
triggerInfoAddress	O	O	M	xs:anyURI	No default When the *triggerPurpose* is "establishConnection", and this field is provided by the trigger originator, then this field shall be configured with an unstructured CSE-Relative-Resource-ID of the <remoteCSE> or <AE> resource of the trigger recipient. The trigger recipient shall update the *pointOfAccess* attribute of this resource.

Field Name	Request Optionality			Data Type	Default Value and Constraints
	Establish Connection	registration Request	execute CRUD		
					When the *triggerPurpose* is "establishConnection", and this field is not provided by the trigger originator, the trigger recipient shall establish a network connection with its Registrar CSE but not update its *pointOfAccess*. When the *triggerPurpose* is "registrationRequest", and this field is provided by the trigger originator, then this field is the unstructured CSE-Relative-Resource-ID of the Registrar CSE's <cseBase> resource that the trigger recipient shall register to. When the *triggerPurpose* is "registrationRequest", and this field is not provided by the trigger originator, the trigger recipient shall register to the Registrar CSE using a pre-provisioned address of the Registrar CSE. The pre-provisioning method is outside the scope of the present document. When the *triggerPurpose* is "executeCRUD", this field is mandatory and shall be configured with an unstructured CSE-Relative-Resource-ID by the trigger originator. The trigger originator shall also specify the type of CRUD operation in the *triggerInfoOperation* field and the type of resource in the *targetedResourceType* field. The trigger recipient shall perform the CRUD operation specified by the *triggerInfoOperation* field on this resource. When the *triggerPurpose* is "enrolmentRequest", this field is mandatory and shall be configured with the absolute URI of the <MEFBase> resource of the MEF that the ASN/MN-CSE or ADN-AE shall enrol to.
triggerInfoPoA	O	O	O	m2m:poaList	No default List of pointOfAccess of the trigger originator. When *triggerInfoAddress* is included, the trigger originator shall configure this field with at least one supported pointOfAccess.
triggerInfoOperation	NP	NP	M	m2m:operation	No default See clause 6.3.4.2.5. When the *triggerPurpose* is

Field Name	Request Optionality			Data Type	Default Value and Constraints
	Establish Connection	registration Request	execute CRUD		
					"executeCRUD", the trigger originator shall configure this field with the CRUD operation to perform on the targeted resource specified by *triggerInfoAddress*.
triggerInfoResourceType	NP	NP	M	m2m: resourceType	No default See clause 6.3.4.2.1. When the *triggerPurpose* is "executeCRUD", the trigger originator shall configure this field with the resource type of the targeted resource specified by *triggerInfoAddress*.
triggerInfoAE-ID	NP	NP	O	m2m:ID	No default This field is included in the payload by the trigger originator when the purpose of the trigger is to request an ASN/MN-AE of the trigger recipient is to perform a CRUD operation. This field identifies the ASN/MN-AE that should perform the CRUD operation. The type of CRUD operation to perform shall be specified by the trigger originator in the *triggerInfoOperation.* The resource to perform the operation on shall be specified by the trigger originator in the *triggerInfoAddress.* The type of resource shall be specified by the trigger originator in the *targetedResourceType*.
triggerInfoSerializationTypes	*O*	*O*	*O*	list of m2m:serializationType	This field may be configured by the trigger originator. The field indicates which types of serializations the trigger originator supports in requests from the trigger recipient (i.e. XML, JSON and/or CBOR). The default value is JSON.

NOTE: Mandatory payload fields are only mandatory if the trigger payload is present.

The trigger payload may be serialized in XML, JSON or CBOR format. The IN-CSE shall serialize the trigger payload based on the *contentSerialization* attribute of the trigger recipient's <AE> or <remoteCSE> resource. If the trigger recipient has not yet registered to the IN-CSE, and the *contentSerialization* attribute of the trigger recipient's <AE> or <remoteCSE> resource is not available to the IN_CSE, the IN-CSE may use any of the supported serialization formats.

Annex A (normative):
Binding Mch to Diameter for Charging

A.1 Introduction

The present clause provides Diameter binding of Mch.

A.2 Diameter Commands on Mch

A.2.1 Accounting Request Command

The ACR command is sent from the Charging Function (CHF included within the SCA CSF) embedded within the M2M IN to the Charging Server using the Mch reference point. This command is used for Event Based requests.

The ACR message format is defined according to the Diameter Base Protocol in IETF RFC 6733 [14] as follows:

```
<ACR> ::= < Diameter Header: 271, REQ, PXY >
          < Session-Id >
          { Origin-Host }
          { Origin-Realm }
          { Destination-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Destination-Host ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Acct-Interim-Interval ]
          [ Accounting-Realtime-Required ]
          [ Origin-State-Id ]
          [ Event-Timestamp ]
        * [ Proxy-Info ]
        * [ Route-Record ]
        * [ AVP ]
```

A.2.2 Accounting Answer Command

The ACA command is sent from the Charging Server to the Charging Function (CHF included within the SCA CSF) embedded within the M2M IN in response to the ACR command and is used to acknowledge reception of the charging data. This command is used for Event Based responses.

The ACA message format is defined according to the Diameter Base Protocol in IETF RFC 6733 [14] as follows:

```
<ACA> ::= < Diameter Header: 271, PXY >
          < Session-Id >
          { Result-Code }
          { Origin-Host }
          { Origin-Realm }
          { Accounting-Record-Type }
          { Accounting-Record-Number }
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ User-Name ]
          [ Accounting-Sub-Session-Id ]
          [ Acct-Session-Id ]
          [ Acct-Multi-Session-Id ]
          [ Error-Message ]
          [ Error-Reporting-Host ]
          [ Failed-AVP ]
```

```
  [ Acct-Interim-Interval ]
  [ Accounting-Realtime-Required ]
  [ Origin-State-Id ]
  [ Event-Timestamp ]
* [ Proxy-Info ]
* [ AVP ]
```

A.3 Mapping of M2M Recorded Information Elements to AVPs

Table A.3-1 describes the mapping of the M2M Recorded Information Elements identified in oneM2M TS-0001 [6] to the Diameter AVPs.

Table A.3-1: Mapping of Recorded M2M Information Elements to Diameter AVPs

M2M Recorded Information Elements	Diameter AVP
M2M Service Subscription Identifier	Subscription-Id
Application Entity ID	Application-Entity-ID
External ID	External-ID
Receiver	Receiver
Originator	Originator
Hosting CSE-ID	Hosting-CSE-ID
Target ID	Target-ID
Protocol Type	Protocol-Type
Request Operation	Request-Operation
Request Headers size	Request-Headers-Size
Request Body size	Request-Body-Size
Response Headers size	Response-Headers-Size
Response Body size	Response-Body-Size
Response Status Code	Response-Status-Code
Time Stamp	M2M-Event-Record-Timestamp
M2M-Event-Record-Tag	Rating-Group
Control Memory Size	Control-Memory-Size
Data Memory Size	Data-Memory-Size
Access Network Identifier	Access-Network-Identifier
Additional Information	AVP
Occupancy	Occupancy
Group Name	Group-Name
maxNrOfMembers	Maximum-Number-Members
currentNrOfMembers	Current-Number-Members
Subgroup Name	Subgroup-Name
M2M-Node-Id	Node-Id
M2M Service Subscription Identifier	Subscription-Id
Application Entity ID	Application-Entity-ID

A.4 Summary of AVPs used

Table A.4-1 lists the Diameter AVPs specifically used for the offline charging interface.

In Table A.4-1, columns "Used in ACR" and "Used in ACA" identify at a protocol level if the AVP is mandatory, optional, or not allowed. When identified as optional here, an AVP may be considered mandatory for certain conditions as identified in Table 12.1.2.2-1 of oneM2M TS-0001 [6].

AVPs defined for oneM2M specific usage are assigned Vendor-Id of 45687. The formats and usage of oneM2M specific AVPs are defined in the present document in clause A.5.

The table contains the following information:

- AVP Name: The name used in Diameter.
- AVP Vendor ID: The entity defining the AVP code in the next column.
- AVP Code: The AVP Code used in the Diameter AVP Header.
- Used in ACR: Indicates if it is mandatory, optional or not used in the ACR command.
- Used in ACA: Indicates if it is mandatory, optional or not used in the ACA command.
- Used in CCR: Not in the present document.
- Used in CCA: Not in the present document.
- AVP Defined: A reference to where this AVP is defined.
- Value Type: The Diameter format of the AVP data as defined in Basic or Derived AVP Data Format.
- AVP Flag Rules: The rules for how the AVP Flags in the AVP Header may be set.
- May Encrypt: Indicates if the AVP may be encrypted or not.

Table A.4-1: Use Of Diameter AVPs

AVP Name	AVP Vendor Id	AVP Code	Used in				Reference	Value Type	AVP Flag rules				
			ACR	ACA	CCR	CCA			Must	May	Should not	Must not	May Encr.
Access-Network-Identifier	45687	1000	O	-	-	-	[A.5.1]	Unsigned32	M	P	-	V	Y
Acct-Application-Id	0	259	O	O	-	-	[5.5.1]	Unsigned32	M	P	-	V	N
Accounting-Record-Number	0	485	M	M	-	-	IETF RFC 6733 [14]	Unsigned32	M	P	-	V	Y
Accounting-Record-Type	0	480	M	M	-	-	[5.4.2]	Enumerated	M	P	-	V	Y
Application-Entity-ID	45687	1001	O	-	-	-	[A.5.2]	UTF8String	M	P	-	V	Y
Control-Memory-Size	45687	1002	O	-	-	-	[A.5.5]	Unsigned32	M	P	-	V	Y
Current-Number-Members	45687	1003	O	-	-	-	[A.5.6]	Unsigned32	M	P	-	V	Y
Data-Memory-Size	45687	1004	O	-	-	-	[A.5.7]	Unsigned32	M	P	-	V	Y
Destination-Host	0	293	O	-	-	-	IETF RFC 6733 [14]	DiamIdent	M	P	-	V	N
Event-Timestamp	0	55	O	O	-	-	IETF RFC 6733 [14]	Time	M	P	-	V	N
External-ID	45687	1005	O	-	-	-	[A.5.8]	UTF8String	M	P	-	V	Y
Group-Name	45687	1006	O	-	-	-	[A.5.9]	UTF8String	M	P	-	V	Y
Hosting-CSE-ID	45687	1007	O	-	-	-	[A.5.10]	UTF8String	M	P	-	V	Y
Originator	45687	1008	M	-	-	-	[A.5.11]	UTF8String	M	P	-	V	Y
Maximum-Number-Members	45687	1009	O	-	-	-	[A.5.12]	Unsigned32	M	P	-	V	Y
M2M-Event-Record-Timestamp	45687	1010	M	-	-	-	[a.5.13]	Time	M	P	-	V	Y
M2M-Information	45687	1011	M	-	-	-	[A.5.14]	Grouped	M	P	-	V	Y
Node-Id	10415	2064	M	-	-	-	3GPP TS 32.299 [31]	UTF8String	V,M	P	-	-	N
Occupancy	45687	1012	O	-	-	-	[A.5.16]	Unsigned32	M	P	-	V	Y
Origin-Host	0	264	M	M	-	-	IETF RFC 6733 [14]	DiamIdent	M	P	-	V	N
Origin-Realm	0	296	M	M	-	-	IETF RFC 6733 [14]	DiamIdent	M	P	-	V	N
Origin-State-Id	0	278	O	O	-	-	IETF RFC 6733 [14]	Unsigned32	M	P	-	V	N
Protocol-Type	45687	1013	O	-	-	-	[A.5.17]	Enumerated	M	P	-	V	Y
Proxy-Info	0	284	O	O	-	-	IETF RFC 6733 [14]	Grouped	M	-	-	P,V	N
Rating-Group	0	432	O	-	-	-	IETF RFC 4006 [32]	Unsigned32	M	P	-	V	Y
Receiver	45687	1014	O	-	-	-	[A.5.19]	UTF8String	M	P	-	V	Y
Request-Body-Size	45687	1015	O	-	-	-	[A.5.20]	Unsigned32	M	P	-	V	Y
Request-Headers-Size	45687	1016	O	-	-	-	[A.5.21]	Unsigned32	M	P	-	V	Y
Request-Operation	45687	1017	O	-	-	-	[A.5.22]	Enumerated	M	P	-	V	Y
Response-Body-Size	45687	1018	O	-	-	-	[A.5.23]	Unsigned32	M	P	-	V	Y
Response-Headers-Size	45687	1019	O	-	-	-	[A.5.24]	Unsigned32	M	P	-	V	Y
Response-Status-Code	45687	1020	O	-	-	-	[A.5.25]	Enumerated	M	P	-	V	Y
Result-Code	0	268	-	M	-	-	IETF RFC 6733 [14]	Unsigned32	M	P	-	V	N
Route-Record	0	282	O	-	-	-	IETF RFC 6733 [14]	DiamIdent	M	-	-	P,V	N
Service-Context-Id	0	461	O	-	-	-	[A.5.26]	Grouped	M	P	-	V	N
Service-Information	10415	873	O	-	-	-	3GPP TS 32.299 [31]	Grouped	M	P	-	V	N
Session-Id	0	263	M	M	-	-	IETF RFC 6733 [14]	UTF8String	M	P	-	V	Y
Subgroup-Name	45687	1021	O	-	-	-	[A.5.28]	UTF8String	M	P	-	V	Y
Subscription-Id	0	443	M	-	-	-	IETF RFC 4006 [32]	Grouped	M	P	-	V	Y
Subscription-Id-Data	0	444	M	-	-	-	IETF RFC 4006 [32]	UTF8String	M	P	-	V	Y
Subscription-Id-Type	0	450	M	-	-	-	IETF RFC 4006 [32]	Enumerated	M	P	-	V	Y
Target-ID	45687	1022	O	-	-	-	[A.5.32]	UTF8String	M	P	-	V	Y

A.5 oneM2M Specific AVP Usage

A.5.1 Access-Network-Identifier AVP

The Access-Network-Identifier AVP (AVP Code 1000) is of type Unsigned32 and identifies the access network associated with the request triggering the M2M Event Record. The IN-CSE detects the link on which a request came from or was sent to and that link maps to a specific Network and locally configured identifier.

A.5.2 Acct-Application-Id AVP

Since the protocol used on Mch is Diameter Accounting, this AVP shall contain the value of 3 as defined in IETF RFC 6733 [14].

A.5.3 Accounting-Record-Type AVP

The Accounting-Record-Type AVP (AVP Code 480) is of type Enumerated and contains the type of accounting record being sent. The following value is currently defined for the Accounting-Record-Type AVP: EVENT_RECORD (value 1) for an Event Based request.

A.5.4 Application-Entity-ID AVP

The Application-Entity-ID AVP (AVP Code 1001) is of type UTF8String and represents the identity of the M2M Application Entity when it is applicable. The format of the AE-ID is specified in clause 6.2.3.

A.5.5 Control-Memory-Size AVP

The Control-Memory-Size AVP (AVP Code 1002) is of type Unsigned32 and represents the storage memory (in bytes) used to store control related information associated with the M2M event record (excludes data storage associated with container related operations).

A.5.6 Current-Number-Members AVP

The Current-Number-Members AVP (AVP Code 1003) is of type Unsigned32 and represents the current number of members in a group as determined by the responses to a request transmitted to a group. This is the same as the attribute "currentNrOfMembers" for the group as described in Table 7.4.13.1-3.

A.5.7 Data-Memory-Size AVP

The Data-Memory-Size AVP (AVP Code 1004) is of type Unsigned32 and represents the storage memory in bytes, where applicable, to store data associated with container related operations.

A.5.8 External-ID AVP

The External-ID AVP (AVP Code 1005) is of type UTF8String and contains the external ID used to communicate over Mcn where applicable. The format is the same as the M2M-Ext-ID in clause 6.2 Addressing.

A.5.9 Group-Name AVP

The Group-Name AVP (AVP Code 1006) is of type UTF8String and identifies the group associated with a request. It shall be included when the IN initiates a fanning operation. This is the same as the attribute "groupName" for the group as described in Table 7.4.13.1-3.

A.5.10 Hosting-CSE-ID AVP

The Hosting-CSE-ID AVP (AVP Code 1007) is of type UTF8String and represents the identity of the Hosting CSE for the request in case the receiver is not the host. The format of the CSE-ID is specified in clause 6.2.3.

A.5.11 Originator AVP

The Originator AVP (AVP Code 1008) is of type UTF8String and identifies the originator (i.e. from party) of the M2M request. This can be any M2M Node with format as per clause 6.2.3.

A.5.12 Maximum-Number-Members AVP

The Maximum-Number-Members AVP (AVP Code 1009) is of type Unsigned32 and represents the maximum number of members of the group for the Create and Update operations. This is the same as the attribute "maxNrOfMembers" for the group as described in Table 7.4.13.1-3.

A.5.13 M2M-Event-Record-Timestamp AVP

The M2M-Event-Record-Timestamp AVP (AVP code 1010) is of type Time and represents the time for recording the M2M event record.

A.5.14 M2M-Information AVP

The M2M-Information AVP (AVP code 1011) is of type Grouped. Its purpose is to allow the transmission of service information elements used for OneM2M specific charging.

It has the following ABNF grammar:

```
M2M-Information :: =  < AVP Header: 1011>
                     [ Application-Entity-ID ]
                     [ External-ID ]
                     [ Receiver ]
                     [ Originator ]
                     [ Hosting-CSE-ID ]
                     [ Target-ID ]
                     [ Protocol-Type ]
                     [ Request-Operation ]
                     [ Request-Headers-Size ]
                     [ Request-Body-Size ]
                     [ Response-Headers-Size ]
                     [ Response-Body-Size ]
                     [ Response-Status-Code ]
                     [ Rating-Group ]
                     [ M2M-Event-Record-Timestamp ]
                     [ Control-Memory-Size ]
                     [ Data-Memory-Size ]
                     [ Access-Network-Identifier ]
                     [ Occupancy ]
                     [ Group-Name ]
                     [ Maximum-Number-Members ]
                     [ Current-Number-Members ]
                     [ Subgroup-Name ][ Node-Id ]
                    * [ AVP ]
```

A.5.15 Node-ID AVP

The Node-Id AVP (AVP Code 2064) is of type UTF8String and includes an optional, operator configurable identifier string for the node generating the Accounting-Record-Number for the Diameter ACR.

A.5.16 Occupancy AVP

The Occupancy AVP (AVP Code 1012) is of type Unsigned32 and represents the overall size (in bytes) of the containers generated by a set of AEs identified by the M2M Service Subscription Identifier.

A.5.17 Protocol-Type AVP

The Protocol-Type AVP (AVP Code 1013) is of type Enumerated and indicates the protocol used for the request. The values are given below:

0	HTTP
1	CoAP
2	MQTT
3	WebSocket
4 .. 99	Reserved for oneM2M defined protocol types
100 .. 199	Operator and vendor specific protocol types

A.5.18 Rating-Group AVP

The Rating-Group AVP (AVP Code 432) is of type Unsigned32 and represents a classification of M2M event records for charging purposes. This is assigned by the IN and is M2M SP specific.

A.5.19 Receiver AVP

The Receiver AVP (AVP Code 1014) is of type UTF8String and identifies the receiver (i.e. to party) of the M2M request. This can be any M2M Node with format as per clause 6.2.3.

A.5.20 Request-Body-Size AVP

The Request-Body-Size AVP (AVP Code 1015) is of type Unsigned32 and represents the number of bytes of the body transported in the Request.

A.5.21 Request-Headers-Size AVP

The Request-Headers-Size AVP (AVP Code 1016) is of type Unsigned32 and represents the number of bytes in the control information header in the Request.

A.5.22 Request-Operation AVP

The Request-Operation AVP (AVP Code 1017) is of type Enumerated and identifies the type of operation requested. The values are defined in Table 6.3.4.2.5-1.

A.5.23 Response-Body-Size AVP

The Response-Body-Size AVP (AVP Code 1018) is of type Unsigned32 and represents the number of bytes of the body transported in the Response.

A.5.24 Response-Headers-Size AVP

The Response-Headers-Size AVP (AVP Code 1019) is of type Unsigned32 and represents the number of bytes in the control information header in the Response.

A.5.25 Response-Status-Code AVP

The Response-Status-Code AVP (AVP Code 1020) is of type Enumerated and identifies the value of returned in the Response Status Code parameter of the Response. The values are defined in clause 6.6.3.

A.5.26 Service-Context-Id AVP

This AVP is of type UTF8String and contains a unique identifier of the Diameter charging specific document that applies the request. This is an identifier allocated by the service provider, by the service element manufacturer, or by a standardization body, and shall uniquely identify a given Diameter charging specific document.

The format of the Service-Context-Id is:

"extensions"."Release"."service-context" "@" "domain"

The OneM2M specific values "service-context" "@" "domain" are:

ts0004@oneM2M.org for OneM2M charging

The "Release" indicates the OneM2M Release the service specific document is based upon e.g. 1 for Release 1.

The "extensions" is operator specific information to any extensions in a service specific document.

A.5.27 Service-Information AVP

The Service-Information AVP (AVP code 873) is of type Grouped. Its purpose is to allow the transmission of additional OneM2M specific information elements.

The complete ABNF syntax is defined and maintained in 3GPP TS 32.299 [31]. The group structure includes zero or more occurrences of the Subscription-Id AVP and the M2M-Information AVP.

The format and content of the M2M-Information AVP which includes the OneM2M specific AVPs are specified in the present document.

A.5.28 Subgroup-Name AVP

The Subgroup-Name AVP (AVP Code 1021) is of type UTF8String and identifies the subgroup associated with a request. It shall be included when the IN initiates a fanning operation and one of the members of the group is a. This is the same as the attribute "groupName" for the subgroup as described in Table 7.4.13.1-3.

A.5.29 Subscription-Id AVP

The Subscription-Id AVP (AVP Code 443) is of type Grouped with structure defined in IETF RFC 4006 [32]. The Subscription-Id AVP includes a Subscription-Id-Data AVP that holds the identifier and a Subscription-Id-Type AVP that defines the identifier type.

For M2M, this identifies the M2M Service Subscription ID associated with the request. This is determined by association maintained by the M2M SP as per clause 12.1.3 in oneM2M TS-0001 [6].

A.5.30 Subscription-Id-Data AVP

The Subscription-Id-Data AVP (AVP Code 444) is of type UTF8String as defined in IETF RFC 4006 [32]. The Subscription-Id-Data is used to identify the M2M Service Subscription. The Subscription-Id-Type AVP defines which type of identifier is used.

A.5.31 Subscription-Id-Type AVP

The Subscription-Id-Type AVP (AVP Code 450) is of type Enumerated as defined in IETF RFC 4006 [32]. It is used to determine which type of identifier is carried by the Subscription-Id AVP. The type(s) to be supported is(are) determined by the M2M SP.

A.5.32 Target-ID AVP

The Target-ID AVP (AVP Code 1022) is of type UTF8String and identifies the target URL for the M2M request if available.

Alternatively the Target-ID AVP can identify the target resource identifier with format defined in clause 6.3.4.

Annex B (normative): 3GPP™ MTC Interworking Device triggering

Refer to oneM2M TS-0026 3GPP Interworking [43].

Annex C (informative):
XML examples

C.1 XML schema for container resource type

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification


The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and
other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with
these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does
it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this
document.
© 2015, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the
appropriate degree of experience
to understand and interpret its contents in accordance with generally accepted engineering or other
professional standards
and applicable regulations. No recommendation as to products or vendors is made or should be
implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR
CONFORMS TO ANY STATUTE,
GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF
MERCHANTABILITY OR FITNESS FOR ANY
PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS.
NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT
PARTNER FOR THIS DOCUMENT,
WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER
INCIDENTAL OR CONSEQUENTIAL DAMAGES.
oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS
DOCUMENT IS AT THE RISK OF THE USER.

-->

<xs:schema targetNamespace="http://www.onem2m.org/xml/protocols"
    xmlns:m2m="http://www.onem2m.org/xml/protocols" elementFormDefault="unqualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="CDT-commonTypes-v3_11_0.xsd"/>
    <xs:include schemaLocation="CDT-contentInstance-v3_11_0.xsd"/>
    <xs:include schemaLocation="CDT-subscription-v3_11_0.xsd"/>
    <xs:include schemaLocation="CDT-semanticDescriptor-v3_11_0.xsd"/>
    <xs:include schemaLocation="CDT-timeSeries-v3_11_0.xsd"/>
    <xs:include schemaLocation="CDT-transaction-v3_11_0.xsd"/>


    <xs:element name="container" substitutionGroup="m2m:sg_announceableResource">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="m2m:announceableResource">
                    <xs:sequence>
                        <!-- Common Attribute, specific to <container>, <contentInstance>, <request>
                            and <delivery> and other resources -->
                        <xs:element name="stateTag" type="xs:nonNegativeInteger"/>
                        <xs:element name="creator" type="m2m:ID" minOccurs="0"/>
                        <!-- Resource Specific Attributes -->
                        <xs:element name="maxNrOfInstances" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="maxByteSize" type="xs:nonNegativeInteger" minOccurs="0"/>
                        <xs:element name="maxInstanceAge" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="currentNrOfInstances" type="xs:nonNegativeInteger"/>
                        <xs:element name="currentByteSize" type="xs:nonNegativeInteger"/>
```

```
                        <xs:element name="locationID" type="xs:anyURI" minOccurs="0"/>
                        <xs:element name="ontologyRef" type="xs:anyURI" minOccurs="0"/>
                        <xs:element name="disableRetrieval" type="xs:boolean" minOccurs="0"/>

                        <!-- Child Resources -->
                        <xs:choice minOccurs="0" maxOccurs="1">
                            <xs:element name="childResource" type="m2m:childResourceRef"
                                minOccurs="1" maxOccurs="unbounded"/>
                            <xs:choice minOccurs="1" maxOccurs="unbounded">
                                <xs:element ref="m2m:contentInstance"/>
                                <xs:element ref="m2m:container"/>
                                <xs:element ref="m2m:subscription"/>
                                <xs:element ref="m2m:semanticDescriptor"/>
                                <xs:element ref="m2m:sg_flexContainerResource"/>
                                <xs:element ref="m2m:timeSeries"/>
                                <xs:element ref="m2m:transaction"/>
                            </xs:choice>
                        </xs:choice>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="containerAnnc" substitutionGroup="m2m:sg_announcedResource">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="m2m:announcedResource">
                    <xs:sequence>
                        <!-- Common Attribute, specific to <container>, <contentInstance>, <request>
                            and <delivery> resources -->
                        <xs:element name="stateTag" type="xs:nonNegativeInteger"/>
                        <!-- Resource Specific Attributes -->
                        <xs:element name="maxNrOfInstances" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="maxByteSize" type="xs:nonNegativeInteger" minOccurs="0"/>
                        <xs:element name="maxInstanceAge" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="currentNrOfInstances" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="currentByteSize" type="xs:nonNegativeInteger"
                            minOccurs="0"/>
                        <xs:element name="locationID" type="xs:anyURI" minOccurs="0"/>
                        <xs:element name="ontologyRef" type="xs:anyURI" minOccurs="0"/>
                        <xs:element name="disableRetrieval" type="xs:boolean" minOccurs="0"/>

                        <!-- Child Resources -->
                        <xs:choice minOccurs="0" maxOccurs="1">
                            <xs:element name="childResource" type="m2m:childResourceRef"
                                minOccurs="1" maxOccurs="unbounded"/>
                            <xs:choice minOccurs="1" maxOccurs="unbounded">
                                <xs:element ref="m2m:contentInstance"/>
                                <xs:element ref="m2m:contentInstanceAnnc"/>
                                <xs:element ref="m2m:container"/>
                                <xs:element ref="m2m:containerAnnc"/>
                                <xs:element ref="m2m:subscription"/>
                                <xs:element ref="m2m:semanticDescriptor"/>
                                <xs:element ref="m2m:semanticDescriptorAnnc"/>
                                <xs:element ref="m2m:sg_flexContainerResource"/>
                                <xs:element ref="m2m:sg_announcedFlexContainerResource"/>
                                <xs:element ref="m2m:timeSeries"/>
                                <xs:element ref="m2m:timeSeriesAnnc"/>
                                <xs:element ref="m2m:transaction"/>
                            </xs:choice>
                        </xs:choice>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

C.2 Container resource that conforms to the Schema given above (see clause C.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:container xmlns:m2m="http://www.onem2m.org/xml/protocols"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
    "http://www.onem2m.org/xml/protocols CDT-container-v3_11_0.xsd"
    resourceName="12xx">
  <resourceType>3</resourceType>
  <resourceID>96719</resourceID>
  <parentID>96734</parentID>
  <creationTime>20141003T112032</creationTime>
  <lastModifiedTime>20141003T112032</lastModifiedTime>
  <labels>label1 label2</labels>
  <accessControlPolicyIDs >
    //example.net/IN_CSEID/93405
  </accessControlPolicyIDs/>
  <expirationTime>20141130T120000</expirationTime>
  <stateTag>0</stateTag>
  <creator>C2345</creator>
  <maxNrOfInstances>5</maxNrOfInstances>
  <maxByteSize>104857600</maxByteSize>
  <maxInstanceAge>3600</maxInstanceAge>
  <currentNrOfInstances>2</currentNrOfInstances>
  <currentByteSize>6</currentByteSize>
  <locationID>//example.net/IN_CSEID/1112</locationID>
  <ontologyRef>http://tempuri.org/ontologies/xyz</ontologyRef>

  <childResource name="ci1234" type="4">mn-cse/12xx/ci1234</childResource>
  <childResource name="ci1235" type="4">mn-cse/12xx/ci1235</childResource>
  <childResource name="sub1" type="23">mn-cse/12xx/sub1</childResource>

</m2m:container>
```

Annex D (normative): <mgmtObj> Resource specializations

D.1 Introduction

The annex defines the structure and procedure for the <mgmtObj> resource specializations. The resource specializations specified in the following clauses of this annex shall be created on the IN-CSE when the management request is performed using external technology specific protocols. The IN-CSE further interacts with the management server to perform management requests towards the managed entity. If the management request is performed solely over the M2M Service Layer, the <mgmtObj> resource specializations are created on the managed entity if the managed entity is equipped with a CSE. If the managed entities are non-oneM2M Nodes, the resources are created on the MN-CSE of the managed entity. The details can be found in the oneM2M TS-0001 [6].

D.2 Resource [firmware]

D.2.1 Introduction

The detailed description of the [firmware] resource can be found in clause D.2 of the oneM2M TS-0001 [6].

Table D.2.1-1: Data Type Definition of [firmware]

Data Type ID	File Name	Note
firmware, firmwareAnnc	CDT-firmware-v3_11_0.xsd	

Table D.2.1-2: Resource specific attributes of [firmware]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1001 (firmware)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
version	M	O	xs:string	
firmwareName	M	O	xs:string	
URL	M	O	xs:anyURI	
update	M	O	xs:boolean	
updateStatus	NP	O	m2m:actionStatus	

D.2.2 Resource specific procedure on CRUD operations

D.2.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.2.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.2.2.2.

May start to download the firmware image from the location indicated by attribute URL in the firmware resource.

D.2.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operations to be performed after Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

If the attribute *update* is present in the request and set to true, download the new firmware image from the address indicated by attribute *URL* of the firmware resource if it is not already downloaded else use the downloaded firmware image to update the current using firmware. The Receiver may need to update the *fwVersion* attribute of the [deviceInfo] resource.

D.2.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.2.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.2.2.2:

Delete the downloaded firmware image locally.

D.3 Resource [software]

D.3.1 Introduction

The detailed description of the [software] resource can be found in clause D.3 of oneM2M TS-0001 [6].

Table D.3.1-1: Data Type Definition of [software]

Data Type ID	File Name	Note
software, softwareAnnc	CDT-software-v3_11_0.xsd	

Table D.3.1-2: Resource specific attributes of [software]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1002 (software)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
version	M	O	xs:string	
softwareName	M	O	xs:string	
URL	M	O	xs:anyURI	
install	NP	O	xs:boolean	
uninstall	NP	O	xs:boolean	
installStatus	NP	NP	m2m:actionStatus	
activate	NP	O	xs:boolean	
deactivate	NP	O	xs:boolean	
activeStatus	NP	NP	m2m:actionStatus	

D.3.2 Resource specific procedure on CRUD operations

D.3.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> resource specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.3.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

May start to download the software package from the location indicated by attribute *URL* in the software resource.

D.3.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operations to be performed after Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

If the attribute *install* is present in the request and set to true, install the new software package downloaded from the address indicated by attribute *URL* of the [software] resource.

When the attribute *uninstall* of the [software] resource is updated to true, uninstall the corresponding software of the [software] resource.

When the attribute *install* and *uninstall* of the [software] resource are simultaneously set to true in request, the CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

When the attribute *activate* of the [software] resource is updated to true, activate the corresponding software of the [software] resource.

When the attribute *deactivate* of the [software] resource is updated to true, deactivate the corresponding software of the [software] resource.

When the attribute *activate* and *deactivate* of the [software] resource are simultaneously set to true in request, the CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

The Receiver may need to update the s*wVersion* attribute of the [deviceInfo] resource.

D.3.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.3.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.2.2.2.

Delete the downloaded software package locally.

D.4 Resource [memory]

D.4.1 Introduction

The detailed description of the [memory] resource can be found in clause D.4 of oneM2M TS-0001 [6].

Table D.4.1-1: Data Type Definition of [memory]

Data Type ID	File Name	Note
memory, memoryAnnc	CDT-memory-v3_11_0.xsd	

Table D.4.1-2: Resource specific attributes of [memory]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1003 (memory)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
memAvailable	M	O	xs:unsignedLong	Unit: Byte
memTotal	M	O	xs:unsignedLong	Unit: Byte

D.4.2 Resource specific procedure on CRUD operations

D.4.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.4.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.4.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.4.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.4.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.5 Resource [areaNwkInfo]

D.5.1 Introduction

The detailed description of the [areaNwkInfo] resource can be found in clause D.5 of oneM2M TS-0001 [6].

Table D.5.1-1: Data Type Definition of [areaNwkInfo]

Data Type ID	File Name	Note
areaNwkInfo, areaNwkInfoAnnc	CDT-areaNwkInfo-v3_11_0.xsd	

Table D.5.1-2: Resource specific attributes of [areaNwkInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1004 (areaNwkInfo)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
areaNwkType	M	O	xs:string	
listOfDevices	O	O	m2m:listOfURIs	

D.5.2 Resource specific procedure on CRUD operations

D.5.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.5.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.5.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.5.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.5.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.6 Resource [areaNwkDeviceInfo]

D.6.1 Introduction

The detailed description of the [areaNwkDeviceInfo] resource can be found in clause D.6 of oneM2M TS-0001 [6].

Table D.6.1-1: Data Type Definition of [areaNwkDeviceInfo]

Data Type ID	File Name	Note
areaNwkDeviceInfo, areaNwkDeviceInfoAnnc	CDT-areaNwkDeviceInfo-v3_11_0.xsd	

Table D.6.1-2: Resource specific attributes of [areaNwkDeviceInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1005 (areaNwkDeviceInfo)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
devID	M	O	xs:string	
devType	M	O	xs:string	
areaNwkId	M	O	xs:anyURI	
sleepInterval	O	O	xs:nonNegativeInteger	Unit: second
sleepDuration	O	O	xs:nonNegativeInteger	Unit: second
devStatus	O	O	xs:string	
listOfNeighbors	O	O	m2m:listOfURIs	

D.6.2 Resource specific procedure on CRUD operations

D.6.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.6.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed".

When the [areaNwkDeviceInfo] is successfully created, the receiver shall update the [areaNwkInfo] resource corresponding to the *areaNwkId* attribute provided in [areaNwkDeviceInfo]. The receiver shall update this [areaNwkInfo] by adding the *resourceID* of the [areaNwkDeviceInfo] to the *listOfDevices* attribute.

D.6.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed".

When the attribute *listOfNeighbors* of the [areaNwkDeviceInfo] resource is updated, the receiver shall modify the corresponding connection relationship among devices in the M2M Area Network by sending signals to non-oneM2M Nodes which is out of scope of oneM2M. According to the response from the non-oneM2M nodes of the modify signal, the receiver shall corresponding update the [areaNwkDeviceInfo] resource which may include the update of the *listOfNeighbors* and the *devType* attribute. The modification may include change of the attach point of the device or removal from the area network.

D.6.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.6.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.7 Resource [battery]

D.7.1 Introduction

The detailed description of the [battery] resource can be found in clause D.7 of oneM2M TS-0001 [6].

Table D.7.1-1: Data Type Definition of [battery]

Data Type ID	File Name	Note
battery, batteryAnnc	CDT-battery-v3_11_0.xsd	

Table D.7.1-2: Resource specific attributes of [battery]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1006 (battery)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
batteryLevel	M	O	xs:unsignedInt	Range: 0-100 Unit: percent
batteryStatus	M	O	m2m:batteryStatus	

D.7.2 Resource specific procedure on CRUD operations

D.7.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.7.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.7.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.7.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.7.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.8 Resource [deviceInfo]

D.8.1 Introduction

The resource [deviceInfo] is used to provide information regarding the device.

The detailed description of the [deviceInfo] resource can be found in clause D.8 of oneM2M TS-0001 [6].

Table D.8.1-1: Data Type Definition of [deviceInfo]

Data Type ID	File Name	Note
deviceInfo, deviceInfoAnnc	CDT-deviceInfo-v3_11_0.xsd	

Table D.8.1-2: Resource specific attributes of [deviceInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1007 (deviceInfo)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
deviceLabel	M	O	xs:string	
manufacturer	M	NP	xs:string	
manufacturerDetailsLink	O	O	xs:string	
manufacturingDate	O	NP	m2m:timestamp	
model	M	NP	xs:string	
subModel	O	NP	xs:string	
deviceType	M	O	xs:string	
deviceName	O	O	xs:string	
fwVersion	O	O	xs:string	
swVersion	O	O	xs:string	
hwVersion	O	NP	xs:string	
osVersion	O	O	xs:string	
country	O	NP	xs:string	
location	O	O	xs:string	
systemTime	O	O	m2m:timestamp	
supportURL	O	O	xs:anyURI	
presentationURL	O	O	xs:anyURI	
protocol	O	O	m2m:protocolList	

D.8.2 Resource specific procedure on CRUD operations

D.8.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.8.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.8.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.8.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.8.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.9 Resource [deviceCapability]

D.9.1 Introduction

The resource [deviceCapability] is used to provide information regarding the device.

The detailed description of the [deviceCapability] resource can be found in clause D.9 of oneM2M TS-0001 [6].

Table D.9.1-1: Data Type Definition of [deviceCapability]

Data Type ID	File Name	Note
deviceCapability, deviceCapabilityAnnc	CDT-deviceCapability-v3_11_0.xsd	

Table D.9.1-2: Resource specific attributes of [deviceCapability]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1008 (deviceCapability)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
capabilityName	M	O	xs:string	
attached	M	O	xs:boolean	1 true: currently attached to the device 2 false: currently detached to the device
capabilityActionStatus	M	O	m2m:actionStatus	The action (i.e. enable, disable) and the related status. See Table 6.3.5.14-1.
currentState	M	O	xs:boolean	• true: the device capability is enabled • false: the device capability is disabled
enable	O	O	xs:boolean	the value of this attribute is always true
disable	O	O	xs:boolean	the value of this attribute is always true

D.9.2 Resource specific procedure on CRUD operations

D.9.2.1 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.9.2.2 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.9.2.3 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

When the attribute *enable* of the [deviceCapability] resource is updated to true, enable the device capability of the [deviceCapability] resource.

When the attribute *disable* of the [deviceCapability] resource is updated to true, disable the device capability of the [deviceCapability] resource.

When the attribute *enable* and *disable* of the [deviceCapability] resource are simultaneously set to true in request, the CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

D.9.2.4 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.9.2.5 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.10 Resource [reboot]

D.10.1 Introduction

The resource [reboot] is used to provide information regarding the device.

The detailed description of the [reboot] resource can be found in clause D.10 of oneM2M TS-0001 [6].

Table D.10.1-1: Data Type Definition of [reboot]

Data Type ID	File Name	Note
reboot, rebootAnnc	CDT-reboot-v3_11_0.xsd	

Table D.10.1-2: Resource specific attributes of [reboot]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1009 (reboot)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
reboot	O	O	xs:boolean	the value of this attribute is always true
factoryReset	O	O	xs:boolean	the value of this attribute is always true

D.10.2 Resource specific procedure on CRUD operations

D.10.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.10.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.10.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

When the attribute *reboot* of the [reboot] resource is updated to true, reboot the corresponding node.

When the attribute *factoryReset* of the [reboot] resource is updated to true, factory reset the corresponding node shall be applied.

When the attribute *reboot* and *factoryReset* of the [reboot] resource are simultaneously set to true in request, the CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

D.10.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.10.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.11 Resource [eventLog]

D.11.1 Introduction

The Resource [eventLog] is used to provide information regarding the device.

The detailed description of the [eventLog] resource can be found in clause D.11 of oneM2M TS-0001 [6].

Table D.11.1-1: Data Type Definition of [eventLog]

Data Type ID	File Name	Note
eventLog, eventLogAnnc	CDT-eventLog-v3_11_0.xsd	

Table D.11.1-2: Resource specific attributes of [eventLog]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1010 (eventLog)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
logTypeId	M	O	m2m:logTypeId	See Table 6.3.4.2.23-1
logData	M	O	xs:string	The content and format of this attribute is out of the present document.
logStatus	M	O	m2m:logStatus	See Table 6.3.4.2.24-1
logStart	O	O	xs:boolean	the value of this attribute is always true
logStop	O	O	xs:boolean	the value of this attribute is always true

D.11.2 Resource specific procedure on CRUD operations

D.11.2.0 Introduction

When management is performed using technology specific protocols, the procedures defined in clause 7.4.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in clause 7.2.2.

D.11.2.1 Create

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.11.2.2 Update

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed":

When the attribute *logStart* of the [eventLog] resource is updated to true, start the logging.

When the attribute *logStop* of the [eventLog] resource is updated to true, stop the logging.

When the attribute *logStart* and *logStop* of the [eventLog] resource are simultaneously set to true in request, the CSE shall reject the request with a ***Response Status Code*** indicating "BAD_REQUEST" error.

D.11.2.3 Retrieve

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.11.2.4 Delete

Originator:

No change from the generic procedures in clause 7.2.2.1.

Receiver:

No change from the generic procedures in clause 7.2.2.2.

D.12 Resource [cmdhPolicy]

D.12.1 Introduction

The resource [cmdhPolicy] represents a set of rules associated with a specific CSE that govern the behaviour of that CSE regarding rejecting, buffering and sending request or response messages via the Mcc reference point.

The detailed description can be found in clause D.12 of oneM2M TS-0001 [6].

Table D.12.1-1: Data Type Definition of [cmdhPolicy]

Data Type ID	File Name	Note
cmdhPolicy	CDT-cmdhPolicy-v3_11_0.xsd	

Note that the optional <subscription> child resources are not used for CMDH policies.

Table D.12.1-2: Resource specific attributes of [cmdhPolicy]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1011 (cmdhPolicy)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
cmdhPolicyName	M	O	xs:string	None
mgmtLink	M	O	m2m:mgmtLinkRef	1 link to [cmdhDefaults] resource instance, 1 or more link(s) to [cmdhLimits] resource instance(s), 1 or more link(s) to [cmdhNetworkAccessRules] resource instance(s), 1 or more link(s) to [cmdhBuffer] resource instance(s)

The Resource Specific Procedure on CRUD Operations as specified in clause 7.4.15 for the generic <mgmtObj> resource type apply.

D.12.2 Resource [activeCmdhPolicy]

The resource [activeCmdhPolicy] provides a link to the currently active set of CMDH policies.

The detailed description can be found in clause D.12.1 of oneM2M TS-0001 [6].

Table D.12.2-1: Data Type Definition of [activeCmdhPolicy]

Data Type ID	File Name	Note
activeCmdPolicy	CDT-activeCmdhPolicy-v3_11_0.xsd	

Table D.12.2-2: Resource specific attributes of [activeCmdhPolicy]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1012 (activeCmdhPolicy)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
activeCmdhPolicyLink	M	O	m2m:ID	The resource ID of the [cmdhPolicy] resource instance containing the CMDH policies that are currently active for the associated CSE.

D.12.3 Resource [cmdhDefaults]

The resource [cmdhDefaults] defines which CMDH related parameters will be used by default when a request or response message contains the *Event Category* parameter but not any other CMDH related parameters and which default *Event Category* parameter shall be used when none is given in the request or response message. The detailed description can be found in clause D.12.2 of oneM2M TS-0001 [6].

Table D.12.3-1: Data Type Definition of [cmdhDefaults]

Data Type ID	File Name	Note
cmdhDefaults	CDT-cmdhDefaults-v3_11_0.xsd	

Table D.12.3-2: Resource specific attributes of [cmdhDefaults]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1013 (cmdhDefaults)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
mgmtLink	M	O	m2m:mgmtLinkRef	1 or more link(s) to [cmdhDefEcValue] resource instance(s); 1 or more link(s) to [cmdhEcDefParamValues] resource instance(s)

D.12.4 Resource [cmdhDefEcValue]

The resource [cmdhDefEcValue] represents a default value for the ***Event Category*** parameter of an incoming request or response message. This default ***Event Category*** becomes applicable when certain conditions are matched which are defined by the other attributes of this resource. The detailed description can be found in clause D.12.3 of oneM2M TS-0001 [6].

Table D.12.4-1: Data Type Definition of [cmdhDefEcValue]

Data Type ID	File Name	Note
cmdhDefEcValue	CDT-cmdhDefEcValue-v3_11_0.xsd	

Table D.12.4-2: Resource specific attributes of [cmdhDefEcValue]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1014 (cmdhDefEcValue)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
order	M	O	xs:positiveInteger	None
defEcValue	M	O	m2m:eventCat	None
requestOrigin	M	O	m2m:listOfM2MID	None
requestContext	O	O	xs:anyType	None
requestContextNotification	O	O	xs:boolean	None
requestCharacteristics	O	O	xs:anyType	None

D.12.5 Resource [cmdhEcDefParamValues]

The resource [cmdhEcDefParamValues] represents a specific set of default values for the CMDH related parameters ***Request Expiration Timestamp***, ***Result Expiration Timestamp***, ***Operational Execution Time***, ***Result Persistence*** and ***Delivery Aggregation*** that are applicable for a given ***Event Category*** if these parameters are not specified in the message. The detailed description can be found in clause D.12.4 of oneM2M TS-0001 [6].

Table D.12.5-1: Data Type Definition of [cmdhEcDefParamValues]

Data Type ID	File Name	Note
cmdhEcDefParamValues	CDT-cmdhEcDefParamValues-v3_11_0.xsd	

Table D.12.5-2: Resource specific attributes of [cmdhEcDefParamValues]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1015 (cmdhEcDefParamValues)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
applicableEventCategory	M	O	list of m2m:eventCatWithDef	Exactly one instance of this [cmdhEcDefParamValues] resource shall be provisioned which contains a value "0" (default) setting for this attribute.
defaultRequestExpTime	M	O	xs:long	-1 means infinity, unit: ms
defaultResultExpTime	M	O	xs:long	-1 means infinity, unit: ms
defaultOpExecTime	M	O	xs:long	-1 means infinity, unit: ms
defaultRespPersistence	M	O	xs:long	-1 means infinity, unit: ms
defaultDelAggregation	M	O	xs:boolean	None

D.12.6 Resource [cmdhLimits]

The resource [cmdhLimits] represents limits for CMDH related parameter values. The detailed description can be found in clause D.12.5 of oneM2M TS-0001 [6].

Table D.12.6-1: Data Type Definition of [cmdhLimits]

Data Type ID	File Name	Note
cmdhLimits	CDT-cmdhLimits-v3_11_0.xsd	

Table D.12.6-2: Resource specific attributes of [cmdhLimits]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1016 (cmdhLimits)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
order	M	O	xs:positiveInteger	None
requestOrigin	M	O	m2m:listOfM2MID	None
requestContext	O	O	xs:anyType	None
requestContextNotification	O	O	xs:boolean	None
requestCharacteristics	O	O	xs:anyType	None
limitsEventCategory	M	O	list of m2m:eventCat	None
limitsRequestExpTime	M	O	m2m:listOfMinMax	-1 means infinity, unit: ms
limitsResultExpTime	M	O	m2m:listOfMinMax	-1 means infinity, unit: ms
limitsOpExecTime	M	O	m2m:listOfMinMax	-1 means infinity, unit: ms
limitsRespPersistence	M	O	m2m:listOfMinMax	-1 means infinity, unit: ms
limitsDelAggregation	M	O	restricted list of xs:boolean	This attribute defines the permitted settings of the ***DeliveryAggregation*** parameter of request primitives. '0' means 'false' '1' means 'true' '0 1' means 'false' or 'true'

D.12.7 Resource [cmdhNetworkAccessRules]

The resource [cmdhNetworkAccessRules] defines the usage of underlying networks for forwarding information to other CSEs during processing of CMDH-related requests in a CSE. The detailed description can be found in clause D.12.6 of oneM2M TS-0001 [6].

Table D.12.7-1: Type Definition of [cmdhNetworkAccessRules]

Data Type ID	File Name	Note
cmdhNetworkAccessRules	CDT-cmdhNetworkAccessRules-v3_11_0.xsd	

Table D.12.7-2: Resource specific attributes of [cmdhNetworkAccessRules]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1017 (cmdhNetworkAccess Rules)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
applicableEventCategories	M	O	list of m2m:eventCatWithDef	Exactly one instance of this [cmdhNetworkAccess Rules] resource shall be provisioned which contains a value "0" (default) setting for this attribute.
mgmtLink	O	O	m2m:mgmtLinkRef	Zero or more links to [cmdhNwAccessRule] resource instance(s)

D.12.8 Resource [cmdhNwAccessRule]

The resource [cmdhNwAccessRule] defines limits in usage of specific underlying networks for forwarding information to other CSEs during processing of CMDH-related requests. The detailed description can be found in clause D.12.7 of oneM2M TS-0001 [6].

Table D.12.8-1: Data Type Definition of [cmdhNwAccessRule]

Data Type ID	File Name	Note
cmdhNwAccessRule	CDT-cmdhNwAccessRule-v3_11_0.xsd	

Table D.12.8-2: Resource specific attributes of [cmdhNwAccessRule]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1018 (cmdhNwAccessRule)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
targetNetwork	M	O	m2m:listOfM2MID	None
minReqVolume	M	O	xs:nonNegativeInteger	Unit: byte
spreadingWaitTime	M	O	xs:nonNegativeInteger	Unit: ms
backOffParameters	M	O	m2m:backOffParameters	Parameters that define how usage of any of the Underlying Networks matching with the targetNetwork attribute of this [cmdhNwAccessRule] resource shall be handled when attempts to use such networks have failed. only for the specified actions. See clause D.12.7 of oneM2M TS-0001 [6].
otherConditions	O	O	xs:anyType	None
mgmtLink	M	O	m2m:mgmtLinkRef	Link to an instance "allowedSchedule" of a <schedule> resource

D.12.9 Resource [cmdhBuffer]

The resource [cmdhBuffer] represents limits in usage of buffers for temporarily storing information that needs to be forwarded to other CSEs during processing of CMDH-related requests in a CSE. The detailed description can be found in clause D.12.8 of oneM2M TS-0001 [6].

Table D.12.9-1: Data Type Definition of [cmdhBuffer]

Data Type ID	File Name	Note
cmdhBuffer	CDT-cmdhBuffer-v3_11_0.xsd	

Table D.12.9-2: Resource specific attributes of [cmdhBuffer]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.4.15	1019 (cmdhBuffer)
objectIDs	O	NP	See clause 7.4.15	
objectPaths	O	NP	See clause 7.4.15	
description	O	O	See clause 7.4.15	
applicableEventCategory	M	O	list of m2m:eventCatWithDef	Exactly one instance of this [cmdhBuffer] resource shall be provisioned which contains a value "0" (default) setting for this attribute.
maxBufferSize	M	O	xs:nonNegativeInteger	Unit: byte
storagePriority	M	O	xs:positiveInteger	The range of storage priority is from 1 to 10.

Annex E (informative): Procedures for accessing resources

E.1 Accessing resources in CSEs – blocking requests

The result of a Request is sent back to the Originator as part of the Response of the Request. This communication mode could result in long blocking times.

The interaction employing blocking involves the following steps in this order:

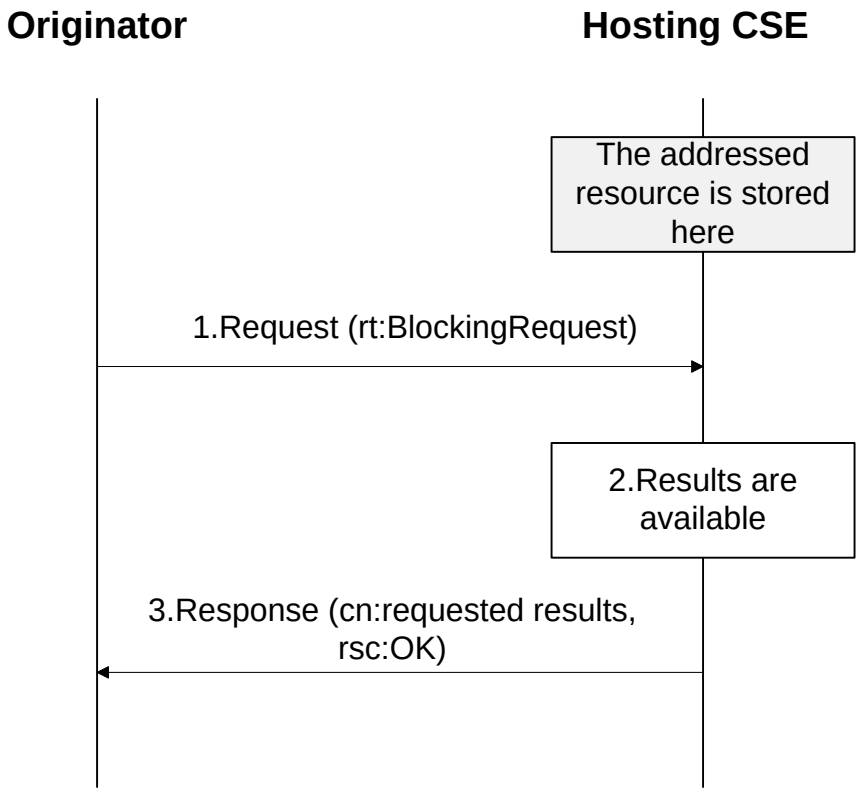

Figure E.1-1: Blocking access to resource

1. The Originator sends a request to access a resource. The ***Response Type*** parameter of the request is set to 'blockingRequest'. The ***Response Type*** parameter can be omitted in this case since 'blockingRequest' is its default value.

2. The Hosting CSE receives the request, and it completes the requested processing of resources.

3. The Hosting CSE responds to Originator, the response contains the requested results in *resource content*, and the ***Response Status Code*** parameter of response is set to "OK".

E.2 Accessing Resources in CSEs - non-blocking requests

E.2.1 Non-blocking models

If the Originator chooses the Blocking mode described in clause E.1, it might have to wait a long time for a response from the Receiver. To avoid this possibility it can choose a Non-Blocking mode. In Non-blocking modes, the Receiver sends an Acknowledgement of the request, which provides a reference to the result of the requested operation. The Originator can retrieve the result at a later time.

There are two forms of Non-blocking mode: Synchronous and Asynchronous.

E.2.2 Synchronous case

The Originator asks for non-Blocking Communication by setting the ***Response Type*** parameter of the Request to 'nonBlockingRequestSynch'. The Receiver CSE responds after acceptance with an Acknowledgement confirming, that it will process the Request further. The Receiver CSE creates a local <request> resource pertaining to the Request received and returns a reference to this created <request> resource as the ***Content*** of the acknowledgement Response. Then the Receiver needs to forward the Request to the next CSE if the Receiver CSE is not the Hosting CSE of the addressed resource. Or the Hosting CSE needs to start handling the Request if the Receiver CSE is the Hosting CSE of the addressed resource.

The Originator of the Request may retrieve the <request> resource afterwards to check on the status of its Request and to inspect the final result of the Request when this is available.

Figure E.2.2-1 illustrates the steps involved in a synchronous non-blocking interaction. In this example the Receiver CSE is the CSE that hosts the resource that is the target of the Originator's request.

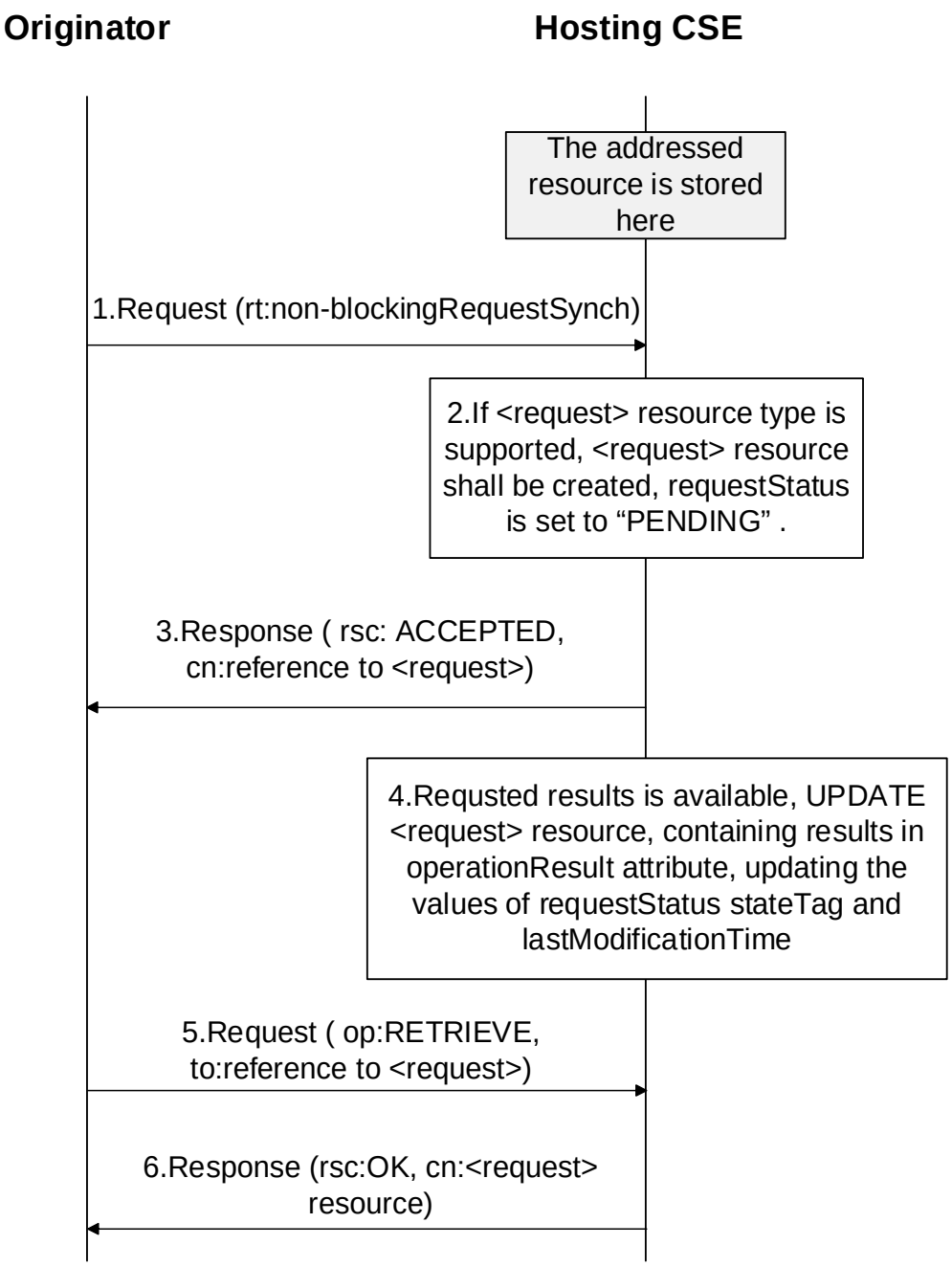

Figure E.2.2-1: Non-Blocking access to resource in synchronous mode (no hop)

1. The originator sends a request to access a resource, setting the ***Response Type*** parameter of request to 'nonblockingRequestSynch'.

2. If the Receiver CSE supports non-blocking synchronous interactions (this is indicated by its support for the <request> resource), it creates an instance of the <request> resource. The *requestStatus* attribute of the <request> resource is set to "PENDING". Refer to Table 7.3.2.2-1 and Table 7.3.2.2-2 for other attributes.

3. The Hosting CSE sends a response to the Originator, the ***Response Status Code*** parameter of its response is set to "ACCEPTED", and a reference to the <request> resource is provided in the ***Content***.

4. The Hosting CSE processes the resource according to the requested operation. When the requested operation has finished, the Hosting CSE will UPDATE the <request> resource, putting the results of the operation into the *operationResult* attribute, and updating the value of *requestStatus* to "COMPLETED", also the values of *stateTag* and *lastModifiedTime*.

5. The Originator requests to RETRIEVE the original requested results by addressing the <request> resource.

6. The Hosting CSE responds to Originator. The response contains the <request> resource as its ***Content***, and the Originator can examine the <request> resource's *requestStatus* attribute to check that the operation has completed and retrieve its results from the *operationResult* attribute.

A variation of synchronous case is depicted in the following clauses. In this variation it is assumed that the addressed resource is not stored in the Registrar CSE, then the Registrar CSE needs to be a Transit CSE to forward the request to the Hosting CSE.

Figure E.2.2-2 illustrates this case.

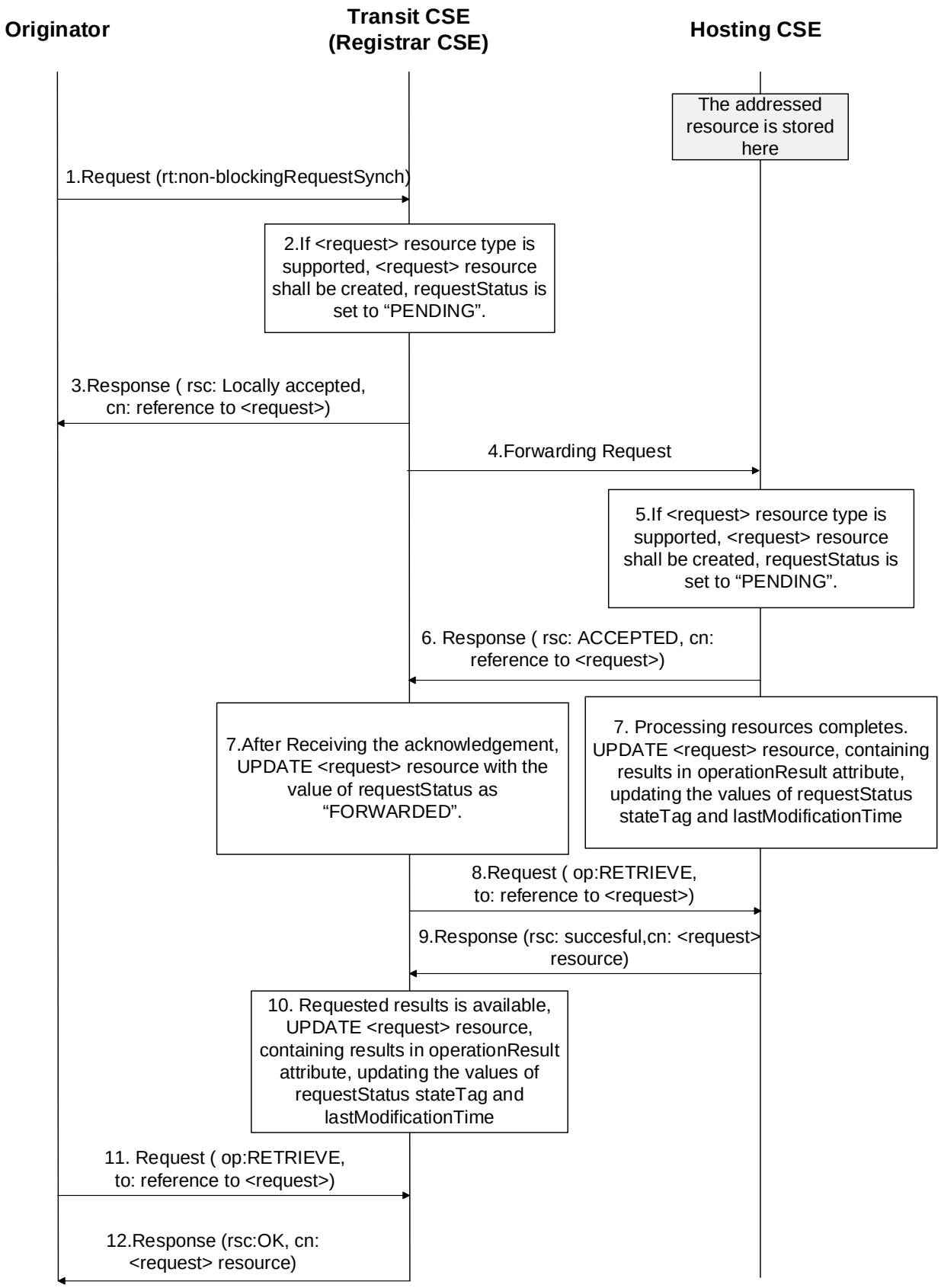

Figure E.2.2-2: Non-Blocking access to resource in synchronous mode (one hop)

1. The Originator sends a request to its Registrar CSE (this is a Transit CSE, not the Hosting CSE), setting the ***Response Type*** parameter of the request to 'nonblockingRequestSynch'.

2. If the Transit CSE supports non-blocking synchronous interactions (this is indicated by its support for the <request> resource), it creates an instance of <request> resource. The *requestStatus* attribute of the <request> resource is set to "PENDING". Refer to Table 7.3.2.2-1 and Table 7.3.2.2-2 for other attributes.

3. The Transit CSE sends a response to the Originator, the ***Response Status Code*** parameter of its response is set to acknowledgement, and a reference to the <request> resource is provided in the ***Content***.

4. The Transit CSE forwards the original request to the Hosting CSE.

5. If the Hosting CSE supports non-blocking synchronous interactions (this is indicated by its support for the <request> resource), it creates an instance of <request> resource. The *requestStatus* attribute of the <request> resource is set to "PENDING". Refer to Table 7.3.2.2-1 and Table 7.3.2.2-2 for other attributes.

6. The Hosting CSE sends a response to the Transit CSE, the ***Response Status Code*** parameter of its response is set to "ACCEPTED" and a reference to the <request> resource is provided in the ***Content***.

7. When Transit CSE receives acknowledgment of this forwarded request, it shall update *requestStatus* attribute of its <request> resource to "FORWARDED". The Hosting CSE processes the resource according to the requested operation. When the requested operation has finished, the Hosting CSE will UPDATE the <request> resource, putting the results of the operation into the *operationResult* attribute, and updating the values of *requestStatus* to "COMPLETED", also the values of *stateTag* and *lastModifiedTime*.

8. The Transit CSE requests to RETRIEVE the original requested results by addressing the <request> resource.

9. The Hosting CSE sends a response to the Transit CSE. The response contains the <request> resource as its ***Content***.

10. The Transit CSE UPDATEs its <request> resource, copying the *operationResult* from the response that it received from the Hosting CSE. It also updates the values of *requestStatus, stateTag* and *lastModifiedTime.*

11. The Originator requests to RETRIEVE the original requested results by addressing the <request> resource.

12. The Transit CSE responds to Originator. The response contains the <request> resource as its ***Content***, and the Originator can examine the <request> resource's *requestStatus* attribute to check that the operation has completed and retrieve its results from the *operationResult* attribute.

Annex F (informative): Guidelines for oneM2M resource type XSD

This annex contains rules to be followed when creating XML Schemas Definition (XSD files to represent the oneM2M resources). The XSD files themselves form part of the oneM2M protocol specification, but the rules used to construct them do not, hence this annex is informative, although it contains normative language.

The purpose of these rules is:

- To keep a consistent style between the schemas for different resources
- To keep the XSD simple
- To allow individual resource schemas to be authored and maintained separately, while minimizing the risk of conflict when they are all used together

1) Each XSD file should include a schema element with following namespace declaration:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.onem2m.org/xml/protocols"
        xmlns:m2m="http://www.onem2m.org/xml/protocols"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        elementFormDefault="unqualified" attributeFormDefault="unqualified" >
```

This defines the prefix xs: for the XML Schema namespace, a target namespace http://www.onem2m.org/xml/protocols, and the prefix m2m: as equivalent for the target namespace. The xsi: namespace can be omitted if the resource has no nillable attributes (see below).
Locally declared elements and attributes shall be unqualified (elementFormDefault and attributeFormDefault declarations are not strictly required since "unqualified" is the default value setting).

2) Each Resource XSD file will contain a Global Element Declaration whose name is the name of the Resource Type in accordance with oneM2M TS-0001 [6]. This means that the root element of a Resource (when represented as an XML instance) contains an m2m: (or equivalent) namespace prefix. If the Resource is announceable, the XSD file will contain a second Global Element Declaration that is used for the Announced variant of the resource. The name of that element will be formed by adding the suffix Annc to the name of the first Global Element. The XSD should not contribute anything to the m2m: namespace other than these root elements.

3) The root element of each resource shall have a required attribute called "resourceName" which gives an identifier for that particular resource instance. A URI to the resource instance can be constructed by taking the URI of its parent and appending /<name> where <name> is the value of the *resourceName* attribute.

4) Each resource attribute of the Resource Type in accordance with TS-0001 Functional Architecture [6] is represented as a child element of the top level element. It shall be declared as an element that is local to the resource that contains it, and so does not have a namespace prefix in any XML instance representation of the resource.

5) Each child resource shall be represented as a child element of the top level element by referring to the global element definition of the child Resource (this allows the child Resource representation to be returned inline). The resource schemas will also include – as an alternative – an element called 'childResource' which is used to return a non-hierarchical URI for the associated child resource, if this has been requested. This element shall have two attributes (in XSD):

   a) type; Data type ID of instances;

   b) name; the name of a child resource instance.

6) Each Resource attribute shall be declared to use one of the following data types:

   a) A data type listed in clause 6.3.2 or 6.3.3.

   b) A list of one of the data types listed in clause 6.3.2 or 6.3.3. If the list type is not already included in clause 6.3.3 it may be defined inside the XSD file for the resource, but if so it will be defined as an anonymous type in the attribute declaration itself.

   c) A data type derived by restriction from one of the types listed in clause 6.3.2 or 6.3.3. This may be added to clause 6.3.3, or defined inside the XSD file for the resource, but in the latter case it will be defined as an anonymous type in the attribute declaration itself.

   d) An anonymous complex type defined as part of the attribute declaration (inside the XSD file for the resource). The complex type should only be composed out of the types listed in clause 6.3.2 or 6.3.3.

7) If a data type is used by more than one attribute (either in the same resource or in two different resources) it will be included in clause 6.3.3, and referenced by each attribute that uses it. Options 6b, 6c, 6d should only be used in cases where the type is only used by one attribute.

8) All Resource types will extend one of the XML complex types described in clause 6.5 and included in the file CDT-commonTypes-v3_11_0.xsd.

9) The resource-specific attributes and child resources shall appear as a sequence of elements in the XSD file, with their order being determined by the order shown in the tables in clause 7.4.

10) Each XSD file shall include an XML comment that contains a oneM2M Copyright Notification Notice of Disclaimer & Limitation of Liability, and a change history. The change history is to be filled in only after the initial release.

11) To enable distinction between element names used for resource attributes and their data types in the m2m: namespace, the use of identical names should be avoided. It is recommended to use the text suffix 'Type' in data type names.

EXAMPLE: <xs:element name="status" type="m2m:statusType />

12) Each mgmtLink shall be represented as a child element 'mgmtLink' which is used to return a non-hierarchical URI for the associated management resource. This element has two attributes (in XSD): a) type; Data type ID of instances, b) name; the name of a child resource instance.

Annex G (normative): Location request

G.1 Introduction

Location Request is a means by which a CSE requests the geographical or physical location information of a target Node to the location server located in the Underlying Network over Mcn reference point. This annex describes only the case of location request when the attribute ***locationSource*** of <locationPolicy> resource type is set to Network Based. Please see clause 7.4.10.

The specific interface used for this request depends on the capabilities of the Underlying Network and other factors. This annex provides the interfaces for location request used for the communication between the CSE and the location server.

G.2 Location request by means of OMA-REST-NetAPI-TerminalLocation interface

G.2.1 Introduction

This OMA REST Network API for Terminal Location specification v1.0 [28] is generally used to open up service capabilities, especially location capability, in the underlying network toward applications. This clause introduces the resources structure and procedures to handle the oneM2M-specified location request. In addition, since this OMA Network API uses only HTTP as underlying message protocol, some binding mapping are mentioned in the procedures in clause G.2.3.

G.2.2 Resource structure of OMA NetAPI for terminal location

When a CSE needs to request the geographical or physical location information of a target CSE or AE hosted in a M2M Node toward a location server located in the Underlying Network over Mcn reference point. The CSE shall request Terminal Location Query following Procedures for Terminal Location (see clause G.2.3).

The OMA REST NetAPI for Terminal Location allows CSE to obtain information about geographical location of a terminal (e.g. Node in oneM2M TS-0001 [6]). In order to obtain location information, CSE shall use one of two services of the Terminal Location API:

- request the current Terminal Location in a single query toward a Location Server;
- subscribe to notifications of periodic Terminal Location updates.

Additionally, in order to track the terminal's movement in relation to the geographic area (circle), crossing in and out (more detail usage is defined in annex E of oneM2M TS-0003 [7]) it is also proposed to use a service of the Terminal Location API:

- subscribe to notification of area updates.

Since oneM2M system utilizes the three services mentioned above, this clause introduces the capabilities that is related to the services from OMA REST NetAPI for Terminal Location [28].

A CSE and a Node shall act as an application and a terminal respectively as described in [28].

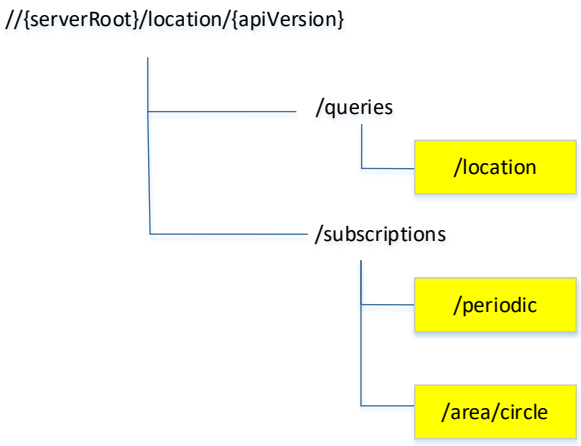

Figure G.2.2-1: Resource Structure defined by NetAPI for Terminal Location

The capabilities used for oneM2M system location request are 'Terminal location'. 'Periodic location notification subscriptions' and 'Area notification subscriptions'. Table G.2.2-1 describes the URL structure, data structure and mapping with CRUD operation of each resource.

Table G.2.2-1: Applicable NetAPI for Terminal Location

Capability	URL Base URL:	Resource Type	Operations			
			C	R	U	D
Terminal location	/location	*TerminalLocation*	no	return current location of the terminal	no	no
Periodic location notification subscriptions	/periodic	*PeriodicNotificationSubscription* (used for CREATE)	create new subscription	return all subscriptions	no	No
Area notification subscription	/area/circle	*CircleNotificationSubscription (used for CREATE)*	create a new subscription	return all subscriptions	No	no

Based on Table G.2.2-1, three resource types, *TerminalLocation*, *PeriodicNotificationSubscription* and *CircleNotificationSubscription* shall be used for the location request specified in the oneM2M system. The resource types are described in the tables below. These tables also contain a column that shows the mapping to attributes in the <locationPolicy> or <accessControlPolicy> resource types. Only attributes that may be utilized by oneM2M system are described. For the detailed information, see [28].

Table G.2.2-2: Resource Type Definition - TerminalLocation

Attributes	OMA NetAPI Defined Type	Description	Relevant Attribute defined by oneM2M
address	xsd:anyURI	Address of the terminal to which the location information applies	*locationTargetID* in the <locationPolicy> resource type
locationRetrievalStatus	common:RetrievalStatus	Status of retrieval for this terminal address.	*locationStatus* in the <locationPolicy> resource type
currentLocation	LocationInfo	Location of terminal.	*content* in the <contentInstance> resource type

Table G.2.2-3: Resource Type Definition - PeriodicNotificationSubscription

Attributes	OMA NetAPI Defined Type	Description	Relevant Attribute defined by oneM2M
address	xsd:anyURI	Addresses of terminals to monitor.	*locationTargetID* in the <locationPolicy> resource type
frequency	xsd:int	Maximum frequency (in seconds) of notifications (can also be considered minimum time between notifications) per subscription.	*locationUpdatePeriod* in the <locationPolicy> resource type
duration	xsd:int	Period of time (in seconds) notifications are provided for. If set to "0" (zero), a default duration time, which is specified by the service policy, will be used. If the parameter is omitted, the notifications will continue until the maximum duration time, which is specified by the service policy, unless the notifications are stopped by deletion of subscription for notifications.	

Table G.2.2-4: Resource Type Definition – CircleNotificationSubscription

Attributes	OMA NetAPI Defined Type	Description	Relevant Attribute defined by oneM2M
latitude	xsd:float	Latitude of center point.	*accessControlLocationRegion* in the <accessControlPolicy> resource type
longitude	xsd:float	Longitude of center point.	*accessControlLocationRegion* in the <accessControlPolicy> resource type
radius	xsd:float	Radius of circle around center point in meters.	*accessControlLocationRegion* in the <accessControlPolicy> resource type
checkImmediate	xsd:boolean	Check location immediately after establishing subscription.	
requester	xsd:anyURI	It identifies the entity that is requesting the information (e.g. 'sip' URI, 'tel' URI, 'acr' URI). The application invokes this operation on behalf of this entity. However, it does not imply that the application has authenticated the requester. If this element is not present, the requesting entity is the application itself. If this element is present, and the requester is not authorized to retrieve location info, a policy exception will be returned.	*authID* in the <locationPolicy> resource type

G.2.3 Procedures for terminal location

G.2.3.1 Request in a single query toward a location server

This procedure shows how to request and return location for a M2M Node.

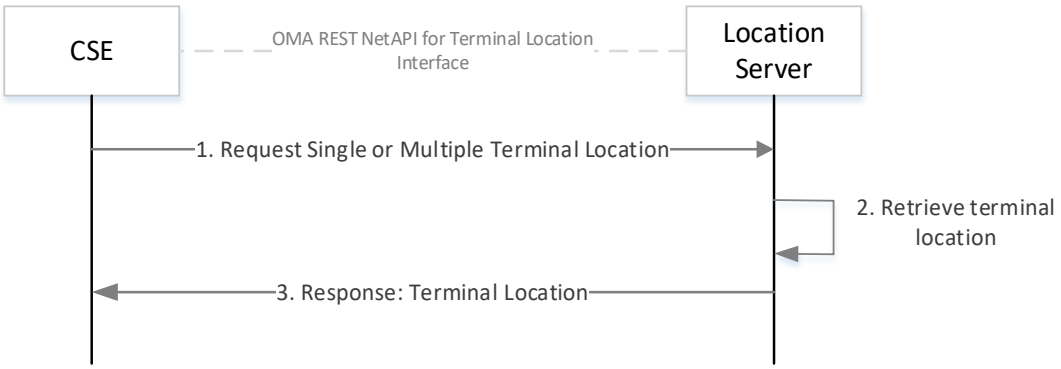

Figure G.2.3.1-1: Single Query Toward Location Server

1. A Hosting CSE requests location for a single terminal (Node) by means of OMA REST NetAPI for terminal location API. This request message shall contain terminal address and Request URL with the address of Location Server using RETRIEVE operation.
   In this step, the *TerminalLocation* resource type described in Table G.2.2-1 shall be used with RETRIEVE operation.

NOTE: GET operation is used for this RETRIEVE operation.

2. The Location Server shall retrieve the location information of the terminal.

3. After the successful retrieve, the Hosting CSE receives the location information.

G.2.4 Subscribe to notifications for periodic location updates

This procedure shows how to control subscriptions for periodic notifications about terminal location.

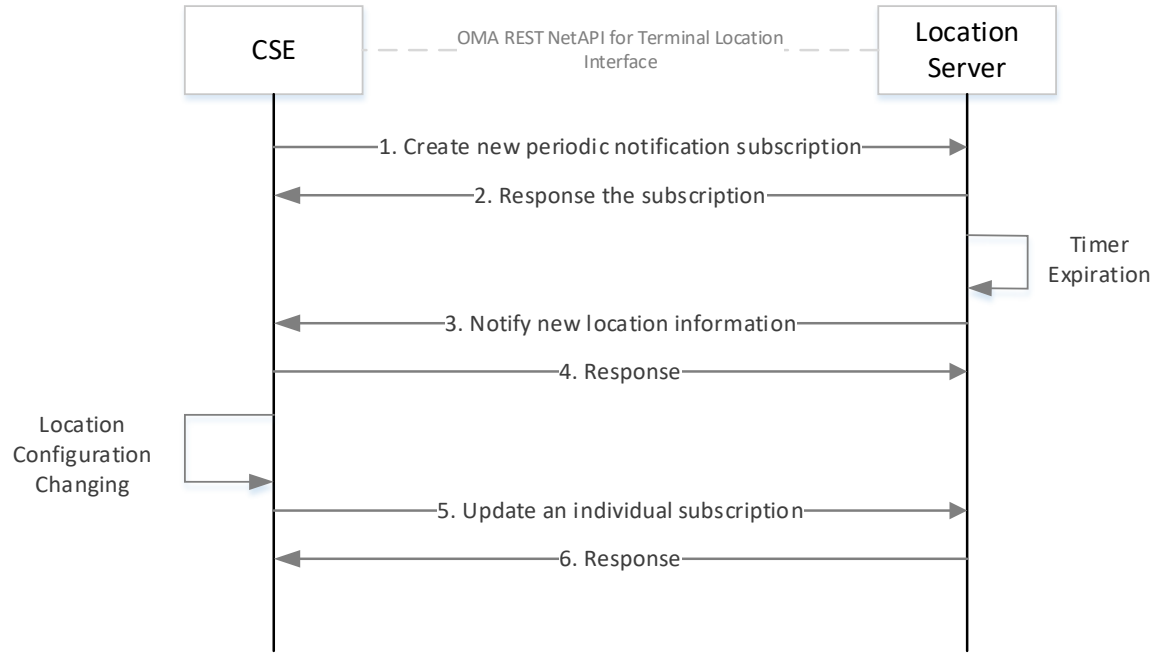

Figure G.2.4-1: Subscribe to Notification for Periodic Location Updates

1. A Hosting CSE shall create a new periodic notification subscription for obtaining location information of a terminal periodically.
   In this step, the PeriodicNotificationSubscription resource type described in Table G.2.2-1 shall be used with CREATE operation.

NOTE 1: POST operation is used for this CREATE operation.

2. After the successful creation of subscription, the Hosting CSE shall receive the response.

3. When the set up timer is expires, the location server shall notify the application of current location information.
   In this step, the notification message shall be used as NOTIFY operation.

NOTE 2: Alternatively, the Hosting CSE obtains the notifications using a Notification Channel [i.3]. This is repeated at specific frequency (periodic information) when the CSE is not reachable.

NOTE 3: POST operation is used for this NOTIFY operation.

4. After the successful receiver of notification, the Hosting CSE shall send a response to the location server.

5. Based upon the location configuration change by the Hosting CSE, it updates an individual subscription for periodic location notification.
   In this step, the PeriodicNotificationSubscription resource type described in the Table G.2.2-1 shall be used with UPDATE operation.

NOTE 4: PUT operation is used for this UPDATE operation.

G.2.5 Subscribe to notifications for area updates

This procedure shows how to subscribe to area update notification.

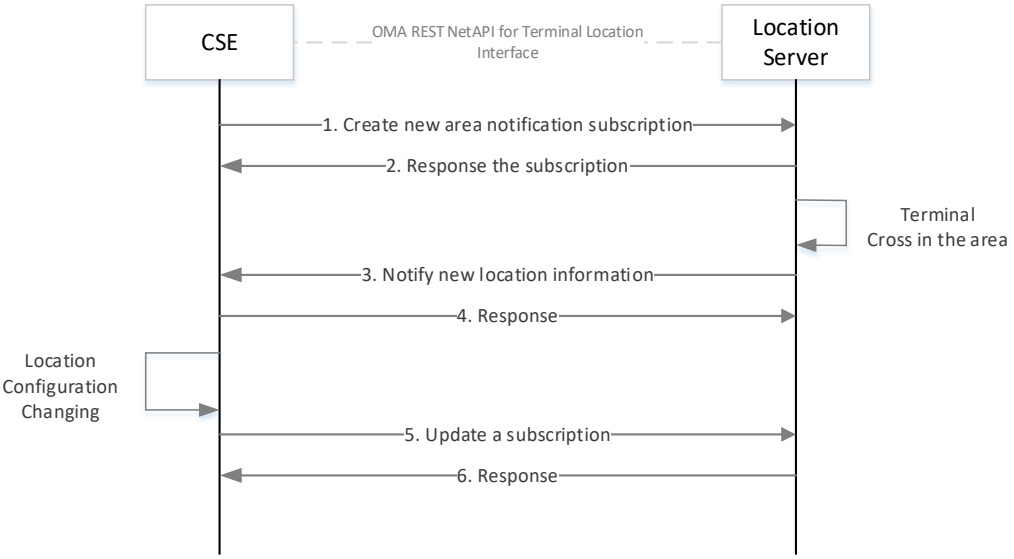

Figure G.2.5-1: Subscribe to Notification for Area Updates

1. A Hosting CSE shall create a new area notification subscription to track the terminal's movement in relation to the geographical area (circle), crossing in and out. In this step, the *CircleNotificationSubscription* resource type described in Table G.2.2-4 shall be used with CREATE operation.

NOTE 1: POST operation is used for this CREATE operation.

2. After the successful creation of subscription, the Hosting CSE shall receive the response.

3. When the target terminal crosses in or out the specified area (circle), the location server shall notify the application of current location information. In this step, the notification message shall be used as NOTIFY operation.

NOTE 2: Alternatively, the Hosting CSE obtains the notifications using a Notification Channel [i.3].

NOTE 3: POST operation is used for this NOTIFY operation.

4. After the successful receiver of notification, the Hosting CSE shall send a response to the location server.

5. Based upon the location configuration change by the Hosting CSE, it updates an individual subscription for area location notification.

   In this step, the *CircleNotificationSubscription* resource type described in Table G.2.2-4 shall be used with UPDATE operation.

NOTE 4: PUT operation is used for this UPDATE operation.

G.3 Location request by means of 3GPP™ MonitoringEvent API

G.3.1 Introduction

This 3GPP MonitoringEvent API for UE location reporting is used to expose location capabilities in the underlying network. The API interface and parameters to handle the oneM2M-specified location request is specified in clause 7.4.7 of oneM2M TS-0026 [43].

Annex H (normative): CMDH message processing

H.1 Pre-requisites

The scope of CMDH processing is to decide at which time and via which communication path to forward request or response messages from a receiver CSE to another CSE. A number of message parameters impact the CMDH processing. CMDH-related request message parameters are:

- ***Event Category*** ('***ec***')
- ***Request Expiration Timestamp*** ('***rqet***')
- ***Result Expiration Timestamp*** ('***rset***')
- ***Operation Execution Time*** ('**oet**')
- ***Result Persistence*** ('***rp***')
- ***Originating Timestamp*** ('**ot**')
- ***Delivery Aggregation*** ('***da***')

CMDH-related response message parameters are:

- ***Event Category*** ('***ec***')**:**
  - ***'ec'*** is needed for response messages as well since response messages can go over multiple hops and CMDH needs to know how to handle them.
- ***Result Expiration Timestamp*** ('***rset***')
- ***Delivery Aggregation*** ('***da***')**:**
  - When a request message was carried inside a <delivery> resource type, also the corresponding response message shall be carried in a <delivery> resource, i.e. the CSE requested to carry out an operation indicated in a request message that reached that CSE via a <delivery> resource, shall also send the response within a <delivery> resource.

The details on how those parameters impact the CMDH processing are described in clause H.2. This annex uses the short names as listed above to refer to request and response parameters.

In the following description it is assumed that the CSE behaviour for CMDH processing is governed by CMDH policies that are represented by [cmdhPolicy] resources and their child resources which are effective for the respective CSE. If legacy device management technologies are used to provision these policies, the information represented by the effective [cmdhPolicy] resources and their child resources may not be available as oneM2M defined resources on the field nodes hosting the respective CSE. This CMDH related policy information may only be available in form of managed objects specific to the used device management technology. In that case the mapping from oneM2M specified [cmdhPolicy] resources and their child resources to equivalent objects of the deployed legacy device management technology shall be used to substitute the respective information contained in [cmdhPolicy] resources and their child resources in the description below. Therefore, whenever reference to [cmdhPolicy] resources, child resources thereof or any attributes of [cmdhPolicy] resources and their children are used in the description of CMDH processing below, they shall be read as a placeholder for the equivalent objects provided by legacy device management technologies on field nodes that are provisioned with such legacy device management technologies.

For a CSE that is processing request or response messages in CMDH, exactly one set of policies represented by a [cmdhPolicy] resource shall be active, as defined by the [activeCmdhPolicy] child resource of the <node> resource that represents the node which hosts the respective CSE. In case of field nodes that are managed via legacy device management technologies, the active CMDH policy can be represented by management objects of that device management technology. For the sake of simplicity, the term 'active [cmdhPolicy]' is used in this and the following clauses to refer to the active CMDH policy information even if no oneM2M specified resources are used to represent CMDH policies. Before any provisioning of CMDH policies has occurred, the 'active [cmdhPolicy]' and its corresponding managed objects defined for legacy device management technologies shall contain the specified default values as described in the [cmdhPolicy] specific procedures and procedures specific for all its child resources. For that reason, it can be assumed that information for an 'active [cmdhPolicy]' is always present on a CMDH capable CSE.

In addition, the active [cmdhPolicy] can have at least one or more [cmdhLimits] child resources and the active [cmdhPolicy] Hosting CSE shall lookup all [cmdhLimits] child resources. If the attribute ***requestContextNotification*** of any of found [cmdhLimits] resources is present and set to true, the CSE shall establish a subscription to the dynamic context information of the CSE defined in *requestContext* attribute of the found [cmdhLimits] as well as subscription to this [cmdhLimits] resource for all AEs corresponding to the AE-ID or an App-ID appearing in the *requestOrigin* attribute. The subscription(s) shall be established when the [cmdhPolicy] is provisioned or pre-provisioned and any of found [cmdhLimits] child resource has the attribute *requestContextNotification* that is set to true. Hence, both this policy establishment and changes of the context information and the [cmdhLimits] resource shall be notified to the respective AEs and the notification shall contain the limits for CMDH related parameter values defined in [cmdhLimits], context information and subscription reference ID. After this, the AEs which received the notification shall send only allowed '***ec***' messages if '***ec***' is specified by the AEs.

H.2 CMDH processing: processing request or response messages requiring the receiver CSE to forward information to another CSE

H.2.1 Applicability of CMDH processing

If a request or response message that is targeting an entity or a resource in the '***to***' parameter that is not among any of:

- the receiver CSE itself;
- an AE registered with the receiver CSE;
- a resource hosted on the receiver CSE,

and if the message is not a response message with an acknowledgement response code, the receiver CSE of that message needs to forward the message to another CSE via CMDH processing, see also the description in clause 7.2.2. *Description of Generic Procedures* of the present document. For forwarding a message to the target CSE indicated by the '***to***' parameter of the message, the receiver CSE shall determine to which CSE the message needs to be forwarded next. In the following clauses this CSE is referred to as the 'next CSE'. CMDH processing shall be carried out as described in the following clauses.

H.2.2 Partitioning of CMDH processing

The CMDH processing consists of two parts:

A. CMDH message validation: This includes message parameter pre-processing, deciding on acceptance for transporting the message, and buffering of messages.
   This procedure defines how incoming request or response messages that need to be forwarded to other CSE(s) shall be pre-processed, how a decision on acceptance of the message for forwarding to another CSE shall be derived and how the messages shall be queued up before the actual forwarding can happen. Details of CMDH validation are defined in clause H.2.3.

B. CMDH message forwarding: This includes selecting buffered messages and communication path for forwarding the message to another CSE.
This procedure defines how to select among the messages buffered for forwarding to other CSEs the ones that need to be transported at a certain time and how to select an appropriate communication path for transporting the message(s). Details of CMDH message forwarding are defined in clause H.2.4.

CMDH message validation (Part A) will be carried out for each incoming new message for which CMDH processing is applicable.

If CMDH message validation is successful, the received message shall be queued up for the CMDH message forwarding process (Part B) including the associated *storagePriority* attribute as defined in the applicable [cmdhBuffer] resource (see details in the CMDH message validation procedure).

If the queued message was a request message and it was done in non-blocking mode then:

- if the Receiver CSE supports the <request> resource type, it shall create a <request> resource representing the pending non-blocking request;
- the Receiver CSE shall send an acknowledgement response message to the entity that sent the request message directly via Mca or Mcc to the receiver CSE indicating the acceptance of the request;
- if the receiver CSE supports the <request> resource type it shall provide a reference to the created <request> resource in the '***cn***' parameter of the response.

After successful forwarding of such a request message, any incoming response message matching with the Request-ID and the Originator in the <request> resource shall be parsed to update the corresponding attributes of the <request> resource. In case a non-blocking synchronous request was forwarded successfully and a response with acknowledgement was received, it is the responsibility of the CSE that forwarded the message to periodically poll the status of the <request> resource created on the next CSE and update the locally created <request> resource accordingly. When the locally created <request> resource expires the Hosting CSE can remove it. Details on <request> resource specific procedures for polling results are defined in clause 7.3.1.4.

If the queued message was a request message and it was done in blocking mode then memorize the open blocking request by storing its Request-ID and Originator and set a timer for a timeout until which a matching response message with the same Request-ID and Originator shall be received by the CSE processing this message. If no matching response is received when the timeout expires, the receiver CSE shall send a response message to the entity that sent the request to the Receiver CSE indicating unsuccessful processing of the request, unless the Receiver CSE and the Originator are the same. If Receiver CSE and Originator are the same, the Originator can decide internally whether to retry forwarding of the message.

If CMDH message validation is not successful, then the received message shall either get ignored – in case the received message is a response message – or a new error response message shall be sent back to the entity that sent the message to the Receiver CSE – in case the received message is a request message and the Originator is not the Receiver CSE. If Receiver CSE and Originator are the same, the Originator can decide internally whether to create a new request message.

The CMDH message forwarding process (Part B) will handle all queued up messages that shall be forwarded to another CSE. This process shall always be carried out when messages are pending for forwarding to another CSE.

The flow of CMDH processing is depicted in Figure H.2.2-1.

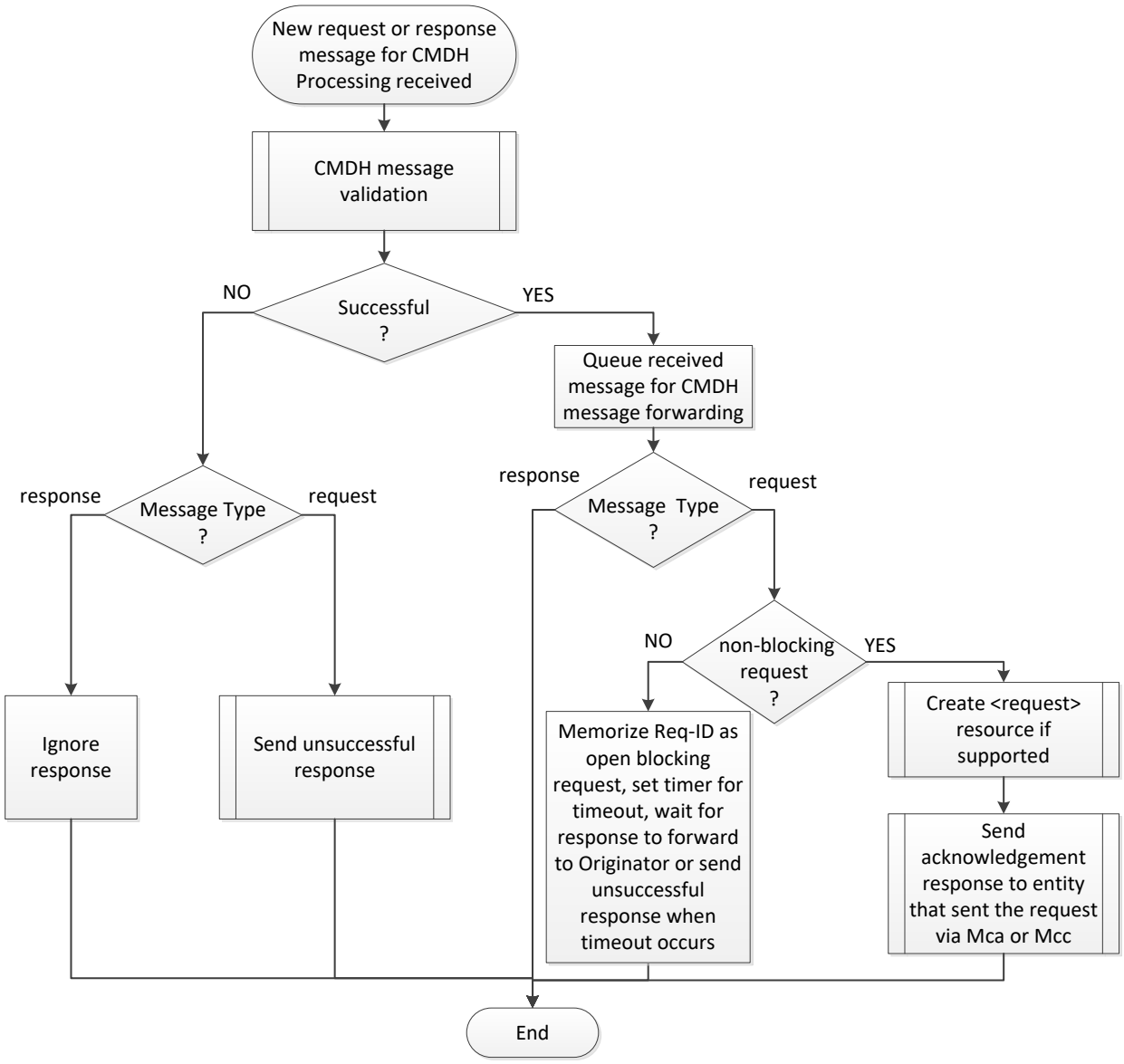

Figure H.2.2-1: CMDH Processing

H.2.3 CMDH message validation procedure

In CMDH message validation, pre-processing of CMDH related parameters of a message for which CMDH-processing applies, deriving the decision on acceptance of a message and the buffering of that messages shall be carried out in line with the following steps. A summary of this processing is depicted in the flow chart at the end of this clause.

1) Filling in missing CMDH-related parameters:

   1.1) Determine the value that shall be used for the '***ec***' parameter of the processed message:

      1.1.1) If the message contains an '***ec***' parameter: Use the value of the '***ec***' parameter provided in the message.

      1.1.2) If the message does not contain an '***ec***' parameter:

         1.1.2.1) Lookup all [cmdhDefEcValue] child resources of the [cmdhDefaults] resource that is a child resource of the provisioned active [cmdhPolicy] resource.

1.1.2.2) If the message is a request message and any of the attributes *requestContext*, and *requestCharacteristics* are present in the found [cmdhDefEcValue] resources, discard all [cmdhDefEcValue] resources from the list of found items for which the context conditions or the request characteristics at time of processing the request message are not met, respectively.

1.1.2.3) Among the remaining found [cmdhDefEcValue] resources do the following selection:

1.1.2.3.1) If present, select the [cmdhDefEcValue] resource containing the AE-ID in the list defined by the *requestOrigin* attribute which matches with the ***'fr'*** parameter in case of a request message or with the ***'to'*** parameter in case of a response message. If multiple [cmdhDefEcValue] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 1.1.2.4.

1.1.2.3.2) If present, select the [cmdhDefEcValue] resource containing the App-ID in the list defined by the *requestOrigin* attribute which matches with the ***'fr'*** parameter in case of a request message or with the ***'to'*** parameter in case of a response message. If multiple [cmdhDefEcValue] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 1.1.2.4.

1.1.2.3.3) If present, select the [cmdhDefEcValue] resource containing the string 'localAE' in the list defined by the *requestOrigin* attribute in case of processing a message where the ***'fr'*** parameter is the *AE-ID* of an AE registered with the CSE processing this message. If multiple [cmdhDefEcValue] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 1.1.2.4.

1.1.2.3.4) If present, select the [cmdhDefEcValue] resource containing the string 'thisCSE' in the list defined by the *requestOrigin* attribute in case of processing a message where the ***'fr'*** parameter is the *CSE-ID* of the CSE processing this message. If multiple [cmdhDefEcValue] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 1.1.2.4.

1.1.2.3.5) Select the [cmdhDefEcValue] resource containing the string 'default' in the list defined by the *requestOrigin* attribute in case of processing a message where no other matches were found.

1.1.2.4) If a [cmdhDefEcValue] resource has been selected in steps 1.1.2.3.1 through 1.1.2.3.4: Use the value of the *defEcValue* attribute of the selected [cmdhDefEcValue] resource as the value for the '***ec***' parameter of the message. Else use the enumeration value of 'bestEffort' for the '***ec***' parameter of the message.

1.2) Filling in values that shall be used for the remaining CMDH-related parameters of messages:

1.2.1) If the message contains any of the CMDH-related parameters '***rqet***', '***rset***', '***oet***', '***rp***': The provided values of the respective parameters in the message shall be used. No filling in is needed for those parameters. If any of the parameters '***rqet***', '***rset***', '***oet***', '***rp***' present in the message is represented in relative time format (i.e. as a duration in units of milliseconds), the receiving CSE shall translate the values of those parameters into absolute time format by adding the duration to the originating timestamp in the '***ot***' parameter of the message. This '***ot***' parameter is an optional message parameter and in case it is not present in a message, it shall be filled in by the first receiving CSE of a message using the time when the message was received.

1.2.2) If the message parameter '***ec***' has a value corresponding to 'bestEffort**'**, use the following values for any missing CMDH-related parameters: For a request message use '***rqet***' = -1('infinite'), '***rset***' = -1 ('infinite'), '***oet***' = 0 ('now'), '***rp***' = 0 ('none'), '***da***' = 'true'. For a response message use '***rset***' = -1 ('infinite'), '***da***' = 'true'. Continue with step 2.

1.2.3) If the message parameter '***ec***' has a value corresponding to 'immediate**'**, do not fill in any remaining missing CMDH-related parameters and continue with step 2.

1.2.4) For any of the missing CMDH-related parameters fill in values as follows:

1.2.4.1) Lookup all [cmdhEcDefParamValues] child resources of the [cmdhDefaults] resource that is a child resource of the provisioned active [cmdhPolicy] resource.

1.2.4.2) Among the found [cmdhEcDefParamValues] resources do the following selection:

1.2.4.2.1) If present, select the [cmdhEcDefParamValues] resource containing the value of the '***ec***' parameter of the message in the list defined by the *applicableEventCategory* attribute. If a match is found, continue processing with step 1.2.4.3.

1.2.4.2.2) Select the [cmdhEcDefParamValues] resource that contains the string *'default'* in the list defined by the *applicableEventCategory* attribute.

1.2.4.3) Use the following attributes of the selected [cmdhEcDefParamValues] resource to fill in any missing CMDH-related message parameters: Fill in the value of the attribute *defaultRequestExpTime* for the parameter '***rqet***' if it is missing. Fill in the value of the attribute *defaultResultExpTime* for the parameter '***rset***' if it is missing. Fill in the value of the attribute *defaultOpExecTime* for the parameter '***oet***' if it is missing. Fill in the value of the attribute *defaultRespPersistence* for the parameter '***rp***' if it is missing. Fill in the value of the attribute *defaultDelAggregation* for the parameter '***da***' if it is missing. Convert the values of '***rqet***', '***rset***', '***oet***' and '***rp***' into absolute time representations if they were filled in during this step, by adding the respective durations to the '***ot***' parameter value. In case where the time duration of the default parameter indicates 'infinity', the absolute time representation of the corresponding primitive parameter shall be set to the largest possible date "99993112T000000".

2) Compare CMDH parameters with allowed CMDH parameter limits:
Check if CMDH-related parameters effective for the message are with allowed limits.

2.1) Lookup all [cmdhLimits] child resources of the provisioned active [cmdhPolicy] resource.

2.2) If the message is a request message and any of the attributes *requestContext,* and *requestCharacteristics* are present in the found [cmdhLimits] resources, discard all [cmdhLimits] resources from the list of found items for which the context conditions or the request characteristics at time of processing the request message are not met, respectively.

2.3) Among the remaining found [cmdhLimits] resources do the following selection:

2.3.1) If present, select the [cmdhLimits] resource(s) containing the AE-ID in the list defined by the *requestOrigin* attribute which matches with the *'**fr**'* parameter in case of a request message or with the *'**to**'* parameter in case of a response message. If multiple [cmdhLimits] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 2.4.

2.3.2) If present, select the [cmdhLimits] resource(s) containing the App-ID in the list defined by the *requestOrigin* attribute which matches with the *'**fr**'* parameter in case of a request message or with the *'**to**'* parameter in case of a response message. If multiple [cmdhLimits] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 2.4.

2.3.3) If present, select the [cmdhLimits] resource(s) containing the string 'localAE' in the list defined by the *requestOrigin* attribute in case of processing a message where the *'**fr**'* parameter is the *AE-ID* of an AE registered with the CSE processing this message. If multiple [cmdhLimits] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 1.1.2.4.

2.3.4) If present, select the [cmdhLimits] resource(s) containing the string 'thisCSE' in the list defined by the *requestOrigin* attribute in case of processing a message where the *'**fr**'* parameter is the *CSE-ID* of the CSE processing this message. If multiple [cmdhLimits] resources match, select the one with the lowest value in the *order* attribute. Continue processing with step 2.4.

2.3.5) Select the [cmdhLimits] resource containing the string 'default' in the list defined by the *requestOrigin* attribute in case of processing a message where no other matches were found.

2.4) Validate if '***ec***' parameter is within allowed range:
If the '***ec***' parameter of the message is not within the list defined by the *limitsEventCategory* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

2.5) Validate if '***rqet***' parameter is within allowed range:
If the '***rqet***' parameter is present in the message and if it is not within the range defined by the '***ot***' parameter and *limitsRequestExpTime* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

2.6) Validate if '***rset***' parameter is within allowed range:
If the '***rset***' parameter is present in the message and if it is not within the range defined by the '***ot***' parameter and *limitsResultExpTime* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

2.7) Validate if '***oet***' parameter is within allowed range:
If the '***oet***' parameter is present in the message and if it is not within the range defined by the '***ot***' parameter and *limitsOpExecTime* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

2.8) Validate if '***rp***' parameter is within allowed range:
If the '***rp***' parameter is present in the message and if it is not within the range defined by the '***ot***' parameter and *limitsRespPersistence* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

2.9) Validate if '***da***' parameter is within allowed range:
If the '***da***' parameter is present in the message and if it is not within the list of allowed values defined by the *limitsDelAggregation* attribute of the selected [cmdhLimits] resource, mark CMDH message validation for this message as not successful and exit CMDH message validation.

3) Check if message complies with network access rules and buffer limits:

3.1) Check if '***ec***' parameter has enumeration value for 'immediate':
If the '***ec***' parameter of the message is 'immediate' bypass any checks on buffering or access network usage rules. Mark the CMDH message validation for this message as successful and end CMDH message validation.

3.2) Check if delivering the message is possible within the boundaries of access network usage rules in CMDH policies:

3.2.1) Lookup all [cmdhNetworkAccessRules] child resources of the provisioned active [cmdhPolicy] resource.

3.2.2) Among the all found [cmdhNetworkAccessRules] resources do the following selection:

3.2.2.1) If present, select the [cmdhNetworkAccessRules] resource containing the value of the '***ec***' parameter of the message in the list defined by the *applicableEventCategory* attribute. If a match is found, continue processing with step 3.2.3.

3.2.2.2) Select the [cmdhNetworkAccessRules] resource that contains the enumeration value for *'default'* in the list defined by the *applicableEventCategory* attribute.

3.2.3) Lookup all [cmdhNwAccessRule] child resources of the selected [cmdhNetworkAccessRules] resource.

3.2.4) Among all found [cmdhNwAccessRule] resources find at least one for which the <schedule> child resource 'allowedSchedule' is allowing usage of the corresponding target network consistent with the '***rqet***' parameter in case of a request message being processed or in line with the '***rset***' parameter in case of a response message being processed. If no matching [cmdhNwAccessRule] resource is found, mark CMDH validation for this message as not successful due to lack of scheduling opportunities and end CMDH message validation. Otherwise continue.

3.3) Check if delivering the message is possible within the boundaries of buffer usage rules in CMDH policies:

3.3.1) Lookup all [cmdhBuffer] child resources of the provisioned active [cmdhPolicy] resource.

3.3.2) Among the all found [cmdhBuffer] resources do the following selection:

3.3.2.1) If present, select the [cmdhBuffer] resource containing the value of the '***ec***' parameter of the message in the list defined by the *applicableEventCategory* attribute. If a match is found, continue processing with step 3.3.3.

3.3.2.2) Select the [cmdhBuffer] resource that contains the enumeration value for *'default'* in the list defined by the *applicableEventCategory* attribute.

3.3.3) Check if the amount of memory needed to buffer the message being validated in addition to the already buffered messages matching with the same buffer usage policy in the selected [cmdhBuffer] resource would exhaust the limit defined by the *maxBufferSize* attribute of the selected [cmdhBuffer] resource or if the available memory for CMDH forwarding on the receiver CSE would get exhausted even when purging buffered messages with lower storage priority.

3.3.3.1) If the check is negative, mark the CMDH message validation for the message being validated as successful, assign the storage priority defined in the *storagePriority* attribute of the selected [cmdhBuffer] resource to the validated message, and end CMDH message validation.

3.3.3.2) If the check is positive, mark the CMDH message validation for the message being validated as not successful and end CMDH message validation.

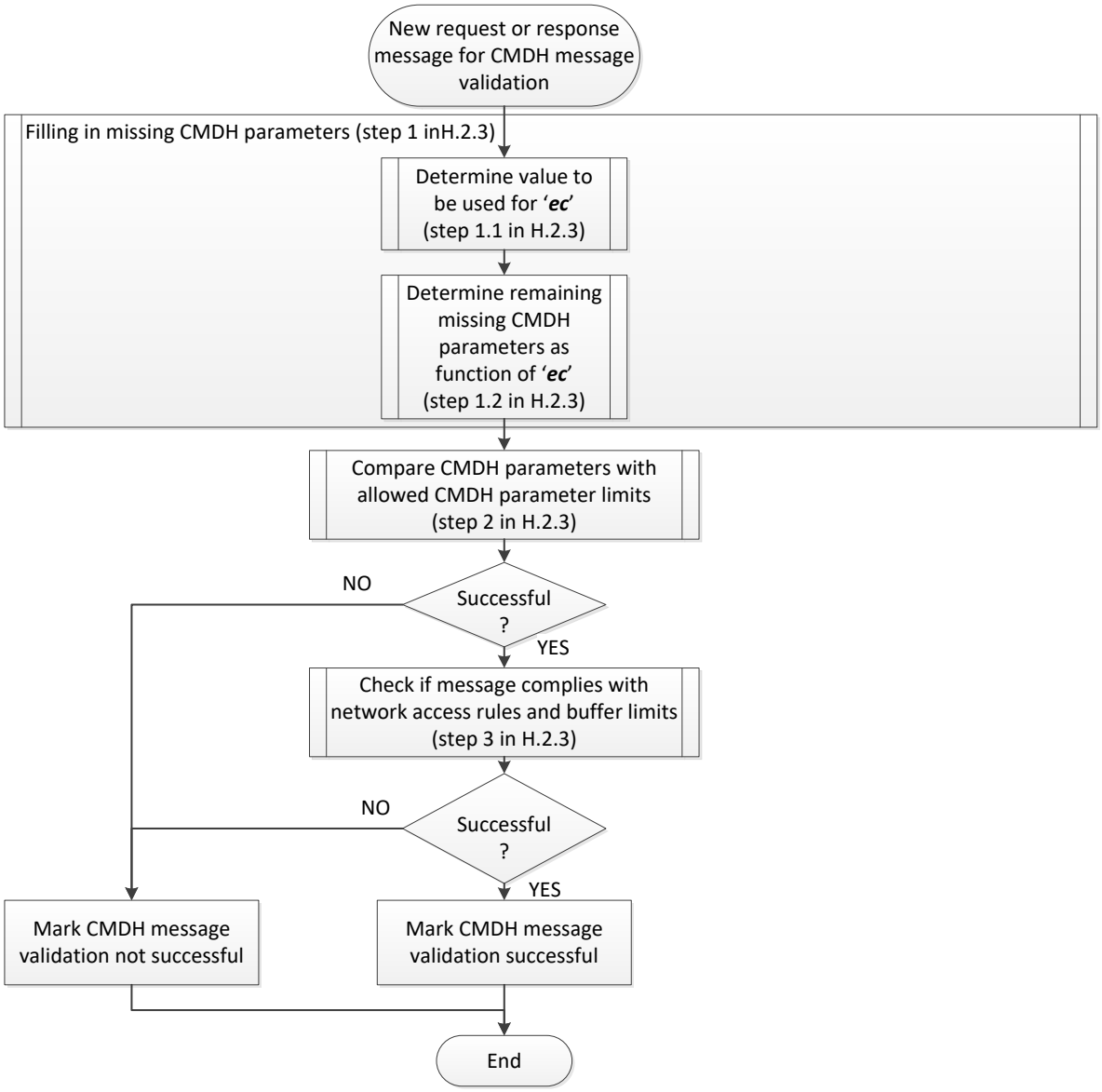

Figure H.2.3-1: CMDH message validation procedure

H.2.4 CMDH message forwarding procedure

The high-level sequence of processing steps for the CMDH message forwarding process is depicted in the flow chart below. Note that this flow chart only represents the reference flow for implementing a standard compliant behaviour. Other standard compliant implementations may be possible as long as the events defined below will result in the same normative message exchanges via reference points.

Occurrence of the following events shall trigger processing in the CMDH message forwarding:

- One or more new message(s) get(s) queued up for CMDH message forwarding.
- Any of the underlying networks becomes usable for message forwarding due to transition(s) in allowed schedule(s) or due to establishing of availability of connectivity (e.g. cable plugged-in, coverage established).
- Any of the underlying networks becomes unusable for message forwarding due to transition(s) in allowed schedule(s) or due to loss of availability of connectivity (e.g. cable unplugged, coverage lost).
- Any message buffered for CMDH forwarding expires.

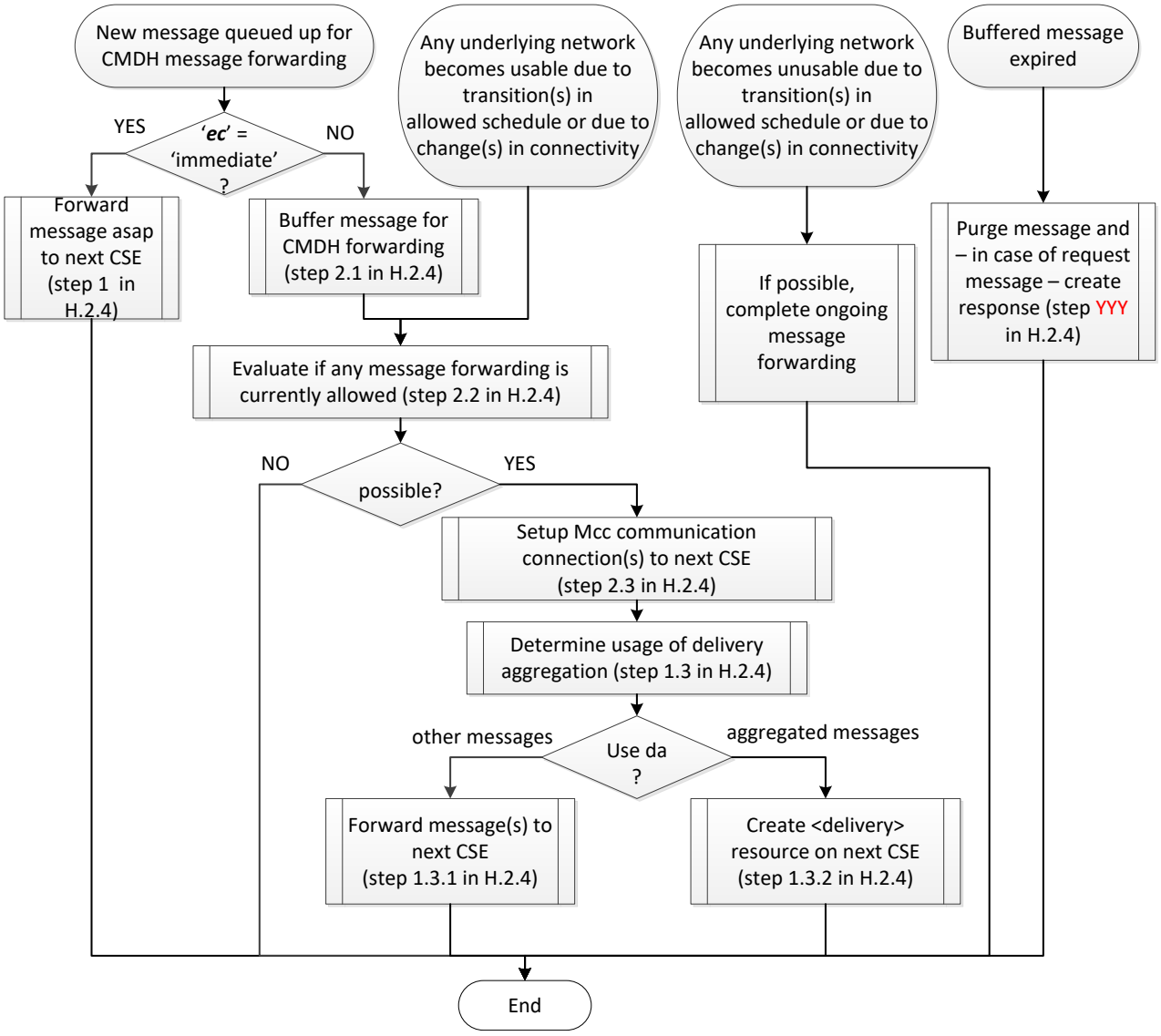

Figure H.2.4-1: CMDH message forwarding procedure

When a new message is being queued up for CMDH message forwarding, carry out the following:

1. If the '***ec***' parameter of the messages has the value 'immediate':

   Forward message as soon as possible to the next CSE. The processing in this situation is described by the flow chart in Figure H.2.4-2.

1.1. If a Mcc communication connection to the next CSE for forwarding the message is already established, continue with step 1.3.

1.2. If no Mcc communication connection to the next CSE for forwarding the message is established pick one underlying network among all underlying networks that can provide communication to the next CSE and establish a Mcc communication connection to the next CSE in line with the rules outlined in clause H.2.5. If establishment of a Mcc communication connection to the next CSE was not successful before the message expires, continue with step 1.4.

1.3. Determine whether delivery aggregation or forwarding of the message itself shall be used:

1.3.1. If the message contains a '***da***' parameter set to the value 'true', the Receiver CSE shall forward this message by creation of a <delivery> resource on the next CSE as outlined in clause 7.4.11. The receiver CSE can combine the forwarded message in the same <delivery> resource with other messages for which the '***da***' parameter set to 'true' and which need to be forwarded to the same target CSE.

1.3.2. If the message is not forwarded using a <delivery> resource, the receiver CSE shall forward the message as is to the next CSE via the established Mcc communication connection.

1.4. If the message could not be forwarded successfully to the next CSE before it expired (e.g. due to repeated unsuccessful attempts to establish a Mcc communication connection or due to the lack of usable underlying networks), the receiver CSE shall carry out the following:

1.4.1. If the message was a response message, ignore the message. End this cycle of CMDH message forwarding and wait for new triggering events.

1.4.2. If the message was a request message:

1.4.2.1. If the request was a blocking request:
Send an error response to the pending blocking request with a matching Request-ID and Originator indicating the reason for failure and close the blocking request. End this cycle of CMDH message forwarding and wait for new triggering events.

1.4.2.2. If the request was a non-blocking request:
Update the associated <request> resource with matching Request-ID and Originator using an error response code indicating the reason for failure. If the non-blocking request was made in asynchronous mode, send a notification with the error response to the notification target(s) of the request. End this cycle of CMDH message forwarding and wait for new triggering events.

1.5. Else, i.e. if the message was forwarded successfully to the next CSE:

1.5.1. If the message was a response and the Receiver CSE has an open blocking request context with a matching Request-ID and matching Originator, mark the open blocking request as closed, end this cycle of CMDH message forwarding and wait for new triggering events.

1.5.2. If the message was a request message:

1.5.2.1. If the request was a blocking request:
Keep the context of the pending blocking request with matching Request-ID and matching Originator open and wait for an incoming response message with the same Request-ID and Originator. End this cycle of CMDH message forwarding and wait for new triggering events.

1.5.3. If the request was a non-blocking request:
Wait for a response to the forwarded request (e.g. response with acknowledgement or error response). Update the associated <request> resource with the matching Request-ID and Originator using a response code that reflects the status of the forwarded request (e.g. accepted by next CSE, unsuccessful). If the next CSE responded with an error response message and the request was in non-blocking asynchronous mode, send a notification request message to the Originator of the forwarded request containing the error response of the next CSE. End this cycle of CMDH message forwarding and wait for new triggering events.

2. Else, i.e. when the '***ec***' parameter of the messages does not have the value corresponding to 'immediate':

2.1.1. Buffer the message to be forwarded in the CMDH forwarding buffer:
The processing in this situation is described by the flow chart in Figure H.2.4.2. If the message is a request message and the '***ec***' parameter of the messages has the value corresponding to 'latest':

2.1.1.1. If the request message is a notification triggered by a subscription:

2.1.1.1.1. Find any buffered request message that is a notification triggered by a subscription with the same subscription reference.

2.1.1.2. Else, i.e. if the request message is not a notification triggered by a subscription:

2.1.1.2.1. Find any buffered request message that has the same values in the ('***fr***', '***to***', '***op***') parameters as the message being processed.

2.1.1.3. If any request message was found in steps 2.1.1.1.1 or 2.1.1.2.1, purge the found message from the CMDH forwarding buffer.

2.1.2. If there is not enough memory available to buffer the message being processed in the CMDH forwarding buffer:

2.1.2.1. Find any buffered messages with storage priority values lower than the one assigned to the message being processed.

2.1.2.2. If any messages are found:
Purge enough messages among the found messages so that the message being processed can be buffered in the CMDH forwarding buffer. Messages which entered the buffer later shall be purged first. In case any request messages need to be purged, carry out the following:

2.1.2.2.1. In case of purging a non-blocking request messages:
Update the associated <request> resource with the same Request-ID as the purged request message with a status indicating unsuccessful completion. If the purged message was made in asynchronous mode, send a response to the notification target(s) of the pending non-blocking request.

2.1.2.2.2. In case of purging a blocking request message:
Send an error response to the open blocking request with the same Request-ID as in the purged request message and close the blocking request.

2.1.2.3. Due to the checking of sufficient memory in CMDH message forwarding buffer during CMDH message validation, there should be enough memory available to accommodate the message to be buffered at this point. If that is still not the case, then do the following:

2.1.2.3.1. In case the message to be buffered is a response message:
Ignore the message to be buffered. End this cycle of CMDH message forwarding and wait for new triggering events.

2.1.2.3.2. In case the message to be buffered is a non-blocking request message:
Update the associated <request> resource with the same Request-ID as the request message to be buffered with a status indicating unsuccessful completion. If the request message to be buffered was made in asynchronous mode, send a response to the notification target(s) of the pending non-blocking request. End this cycle of CMDH message forwarding and wait for new triggering events.

2.1.2.3.3. In case the message to be buffered is a blocking request message:
Respond with an error response message to the open blocking request with the same Request-ID as in the request message to be buffered and close the blocking request. End this cycle of CMDH message forwarding and wait for new triggering events.

2.1.3. Store the message to be buffered with its assigned storage priority in the CMDH forwarding buffer. Include it in future evaluations for possible message forwarding.

2.2. Evaluate if any message forwarding is currently allowed:

2.2.1. For all buffered messages that are pending in CMDH message forwarding carry out the following evaluation steps:

2.2.1.1. Among all [cmdhNetworkAccessRules] child resources of the provisioned active [cmdhPolicy] resource do the following selection:

2.2.1.1.1. If present, select the [cmdhNetworkAccessRules] resource containing a value in the list defined by the *applicableEventCategory* attribute that is equal to the value of the '***ec***' parameter of the buffered message to be evaluated for forwarding. If a match is found, continue processing with step 2.2.1.2.

2.2.1.1.2. Select the [cmdhNetworkAccessRules] resource that contains the string *'default'* in the list defined by the *applicableEventCategory* attribute.

2.2.1.2. Lookup all [cmdhNwAccessRule] child resources of the selected [cmdhNetworkAccessRules] resource.

2.2.1.3. If the attribute *otherConditions* is present in any of the found [cmdhNwAccessRule] resources, discard all [cmdhNwAccessRule] resources from the list of found items for which the conditions expressed by *otherConditions* at time of evaluation of the message for forwarding are not met, respectively.

2.2.1.4. Among the all remaining found [cmdhNwAccessRule] resources find those for which
- the <schedule> child resource allowedSchedule is currently allowing usage of the corresponding target network, and
- the corresponding target network could be used to reach the next CSE for forwarding the message under evaluation.

2.2.1.5.
If any allowed target network was found, memorize the message under evaluation as an allowed message and the allowed target network(s) for the message under evaluation and continue with the next evaluation of buffered messages

2.2.2. When all buffered messages have been evaluated, remove from the memorized list of allowed messages and their allowed target networks those target networks where the amount of data to be forwarded – accumulated over all allowed messages of the same event category – is less than the amount of data indicated in the *minReqVolume* attribute of the corresponding [cmdhNwAccessRule] resource.

2.2.3. Remove any messages from the list of allowed messages for forwarding if no allowed target network is left for that message after the previous step.

2.3. Process messages allowed for forwarding to the next CSE:
If any messages can be forwarded, i.e. if any evaluation of step 2.2 was positive, apply the following steps:

2.3.1. Reuse already established Mcc communication connections or – if needed – establish new Mcc communication connection(s) so that all the messages that are allowed to be forwarded to their next CSE can be forwarded. Some messages may be allowed on the same target network. Follow the procedure outlined in clause H.2.5 for setting up a Mcc communication connection to another CSE via a particular target network. If no usable Mcc communication connection could be established for forwarding a particular allowed message before the message expires, execute step 1.4 in this clause for that message.

2.3.2. For all messages allowed for forwarding and for which Mcc communication connections are established, apply steps 1.3 through 1.5 in this clause.

2.4. Else, i.e. currently no message forwarding is allowed:
End this cycle of CMDH message forwarding and wait for new triggering events.

When any of the underlying networks becomes usable for message forwarding due to transition(s) in allowed schedule(s) or due to establishing of availability of connectivity (e.g. cable plugged-in, coverage established), carry out the processing above in this clause starting with step 2.2.

When any of the underlying networks becomes unusable for message forwarding due to transition(s) in allowed schedule(s) or due to loss of availability of connectivity (e.g. cable unplugged, coverage lost), complete – if at all possible – any ongoing message forwarding procedures. End this cycle of CMDH message forwarding and wait for new triggering events.

When any message buffered for CMDH forwarding expires, carry out step 1.4 in this clause above. End this cycle of CMDH message forwarding and wait for new triggering events.

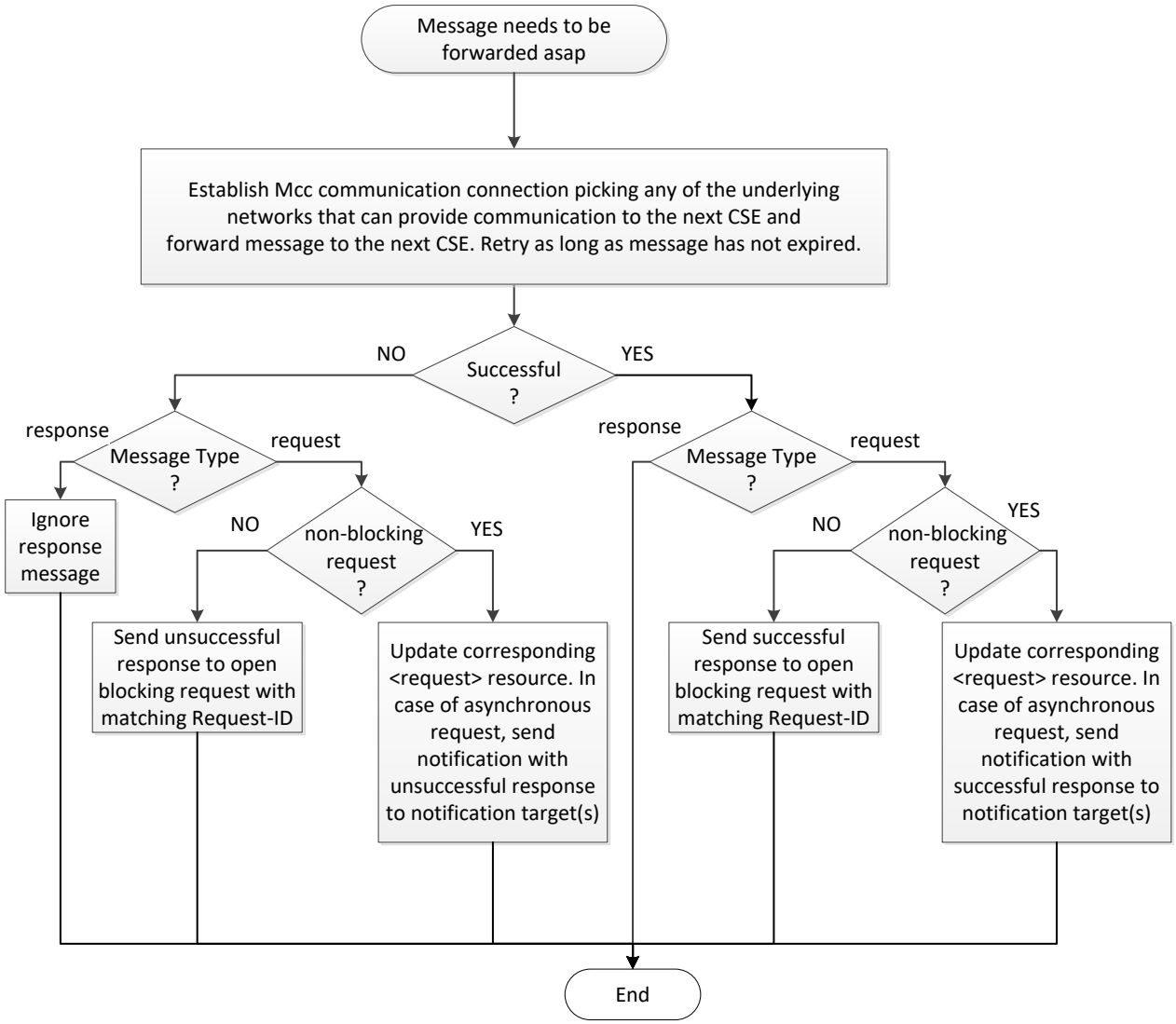

Figure H.2.4-2: Forwarding of messages with '*ec*' = 'immediate'

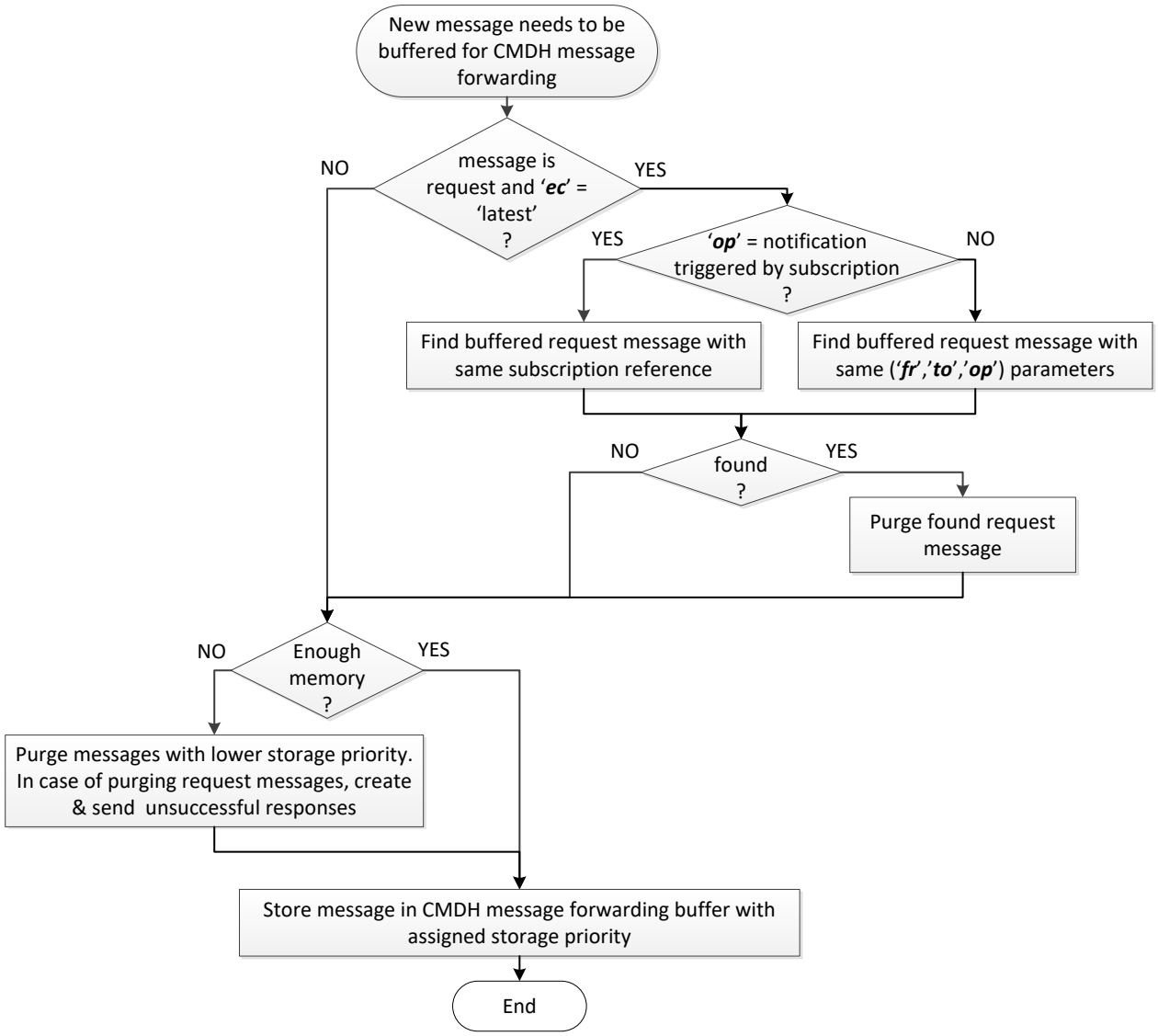

Figure H.2.4-3: Buffering of messages for CMDH message forwarding

H.2.5 Establishment of Mcc communication connection to another CSE

When a Mcc communication connection shall be established via a specific target network for forwarding a message of a specific event category indicated by the '***ec***' parameter of the message, the process of establishing the Mcc communication connection shall be governed by values contained in the *backOffParameters* attribute of the [cmdhNwAccessRule] resource that was used to evaluate whether the message was allowed to be forwarded, as defined in step 2.2 in the procedure outlined in clause H.2.4.

When connectivity via the selected target network to reach the next CSE has not already been established for other reasons, then the CSE that is trying to forward a message buffered for CMDH message forwarding shall establish a new Mcc communication connection via the selected target network for transporting oneM2M messages to the next CSE via a new Mcc instance. This communication connection shall be established following the procedures for authentication and security association using TLS or DTLS as defined in the oneM2M TS-0003 [7] taking into account provisioned security settings. The protocol mapping for transporting oneM2M specified messages via this instance of Mcc shall be selected according to the capabilities of the two end-points of the Mcc instance.

If establishing the Mcc communication connection via the selected target network fails, a new attempt to establish that communication connection shall only be made after waiting for a back-off time according to the value given in the 'back-off time' component of the *backOffParameters* attribute of the effective [cmdhNwAccessRule] resource.

When establishing the Mcc communication connection via the selected target network still fails, for each subsequent new attempt to establish the Mcc communication connection without any successful attempts in-between, the back-off time shall be increased by the value given in the 'back-off time increment' component of the *backOffParameters* attribute of the effective [cmdhNwAccessRule] resource.

The back-off time for waiting before making any new attempt to establish the Mcc communication connection via the selected target network shall not exceed the value given by the 'maximum back-off time' component of the *backOffParameters* attribute of the effective [cmdhNwAccessRule] resource.

When the next CSE is hosted on a node for which a usable Mcc communication connection for forwarding a message to the next CSE can only be established by the next CSE itself, device triggering mechanisms as defined in the oneM2M TS-0001 [6] shall be used.

In case the next CSE can only be reached via communication connections originating from the node that hosts the next CSE, while it is capable of processing incoming oneM2M messages, it is assumed that such a CSE establishes a polling channel as defined in the oneM2M TS-0001 [6] in order to effectively receive unsolicited oneM2M messages.

Annex I (informative): Guidelines for using XSD files in AE and CSE code

I.1 Usage of the oneM2M developed XSD files

The primary purpose of the XSD files developed by oneM2M is described in clause 6.1. This informative annex provides an example of potential usage of the XSD in practical implementations of oneM2M entities (AE and CSE).

As has been specified in clause 8, to enable efficient communication, the short names introduced in clause 8.2 are used in XML and JSON serializations of request and response primitives to identify primitive parameters, and to identify resource names, resource attribute names and their complex data type members when included in the ***Content*** primitive parameter. This implies that short names are applied in any communication over the Mca, Mcc and Mcc' reference points. Nevertheless, the XSD files included in the present release employ the long names for primitive parameters and any other XML elements and attributes.

This annex provides a possible use case of the oneM2M developed XSD files for information.

I.2 Example AE/CSE implementation featuring mapping between short and long names for XML serialization

Figure I.2-1 shows an example where the oneM2M defined XSD files are used as input to a code generator. Such code generators are available for most object-oriented programming languages such as e.g. Java, C++ and Python. The following descriptions include some code examples given in Python syntax. However, corresponding expressions in C++ or Java look very similar.

Code generators generate a library of XSD binding classes corresponding to each of the data types defined in the input XSD files. This library can then be imported into the source code of the respective programming language which implements an AE or CSE.

For example, if this library is denoted schemaLib, instances of a request primitive and of a resource type <contentInstance>, denoted in the Python source code fragment below as reqPrimInstance (internal representation of m2m:requestPrimitive) and contentInstance1 (as internal representation of m2m:contentInstance), respectively, can simply be generated as follows:

```
    import schemaLib
  …
  reqPrimInstance = schemaLib.requestPrimitive()
  contentInstance1 = schemaLib.contentInstance()
```

Each of the instances created in this way represents a data object reflecting the same tree structure as defined in the XSD files that served as input to the code generator.

Any request primitive parameter in reqPrimInstance as defined above can be addressed and assigned values as follows:

```
  reqPrimInstance.operation = operation     #e.g. operation = 1 for CREATE
  reqPrimInstance.to = path                 #path = address of target resource
 reqPrimInstance.from_ = originator  #originator=identifier representing the originator
 reqPrimInstance.requestIdentifier = str(requestIDCounter) #counter in string format
 reqPrimInstance.resourceType = resourceType #e.g. resourceType = 4 for <contentInstance>
```

Parameters defined as complex type in the XSD such as e.g. the *Filter Criteria* primitive parameter can be assigned values as follows:

```
    reqPrimInstance.filterCriteria.createdBefore = '20161201T000000'
    reqPrimInstance.filterCriteria.createdAfter = '20150501T123000'
  reqPrimInstance.filterCriteria.labels = 'label1 label2 label3'
  reqPrimInstance.filterCriteria.attribute.append(pyxb.BIND())
  reqPrimInstance.filterCriteria.attribute[0] = schemaLib.attribute("name0","value0")
```

Note that the class attribute names in the source code are identical with the XML element or attribute names as used in the XSD files (sometimes minor exceptions can occur, for instance in case that a name used in the XSD represents a reserved name in the source code. In such case the code generator typically would append a special suffix to the name, e.g. "_"). Since the XSD uses the long parameter and resource attribute names, these also appear as class attributes in the source code. From an implementation perspective, this is preferable compared to using short names. Using short names in the XSD would result in short names in the source code. However, these short names have essentially lost their semantics and are therefore more difficult do memorize. Any misspelling in the code may easily result in another well-defined short name such that identifying errors in the source code becomes more difficult.

A code generator as considered here, typically also provides a set of class methods and utility functions which allow to generate code objects from a given XML representation, and inversely, to generate XML representations from a code object. For example, consider that a string variable, denote reqPrimXML represents a serialized request primitive as follows (note that this representation corresponds to the example given in clause 8.3.2 but with long names used here):

```
reqPrimXML =
   '<?xml version="1.0" encoding="UTF-8"?>
   <m2m:requestPrimitive xmlns:m2m="http://www.onem2m.org/xml/protocols"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.onem2m.org/xml/protocols CDT-requestPrimitive-v3_11_0.xsd">
      <operation>1</operation>
      <to>//cse1.mym2msp.org/</to>
      <from>//cse1234/app567</from>
      <requestIdentifier>0002bf63</requestIdentifier >
      <resourceType>4</resourceType>
      <primitiveContent>
          <contentInstance resourceName="temp754">
             <contentInfo>application/xml:1</contentInfo>
             <content>PHRpbWU+MTc4ODkzMDk8L3RpbWU+PHRlbXA+MjA8L3RlbXA+DQo=</content>
          </contentInstance>
      </ primitiveContent>
  </m2m:requestPrimitive>'
```

Assuming that the auto-generated library schemaLib includes a utility function createFromDocument(), the following code statement creates an instance reqPrimInstance from the XML serialized request primitive in the string variable reqPrimXML:

```
reqPrimInstance = schemaLib.createFromDocument(reqPrimXML)
```

The root element of the XML string (i.e. m2m:requestPrimitive in this example) identifies the template (class) that need to be used to create the data object reqPrimInstance. All value settings of the parameters are taken from the XML string, e.g. reqPrimInstance.operation is set to 1.

The reverse operation, i.e. generation of an XML string from the data object reqPrimInstance is typically possible with a class method toxml() as follows:

```
reqPrimXML = reqPrimInstance.toxml()
```

If any value settings of reqPrimInstance have not been changed in the given code, the above statement generates the same XML string as given above. Both operations, createFromDocument() and toxml(), also allow to verify the compliance of the XML representations with the XSD that was used as input when generating the schemaLib source code.

The question arises, if there is a way to generate XML or JSON representations that include the short names as defined in clause 8.2 when employing XSD with the long names as described above.

The following outlines two possible ways to resolve this issue.

The first straightforward approach is to use a text parser which replaces the long names used in XML or JSON strings with their corresponding short names, or vice-versa. It is denoted such functions as map_L2S() and map_S2L(). This approach is illustrated in the box labelled "AE or CSE source code" in Figure I.2-1 for an XML serialized string.

Given a string reqPrimXML representing an XML serialized request primitive with long names as described above, the statement:

```
reqPrimXML_sh = map_L2S(reqPrimXML)
```

would produce an XML string that includes the short names as shown in the representation already given in clause 8.3.2.

The reverse operation, generating an XML representation with long names from a representation with short names could be done with:

```
reqPrimXML = map_S2L(reqPrimXML_sh)
```

Both mapping functions require a mapping table which includes all long names and their associated short names. The required mapping table can be derived from Tables 8.2.2-1, 8.2.2-2, 8.2.3-1 to 8.2.3-6, 8.2.4-1 and 8.2.5-1.

In order to work in both mapping directions, the mapping table represents a one-to-one relationship between short and long names.

The second approach is essentially a code-optimized variant of the above first approach.

The source code of the described createFromDocument() and toxml() functions could be extended by the programmer by including the functionality of map_S2L() directly into createFromDocument() and including the functionality of map_L2S() directly into toxml(). An additional function argument could be included which allows to enable and disable the mapping function.

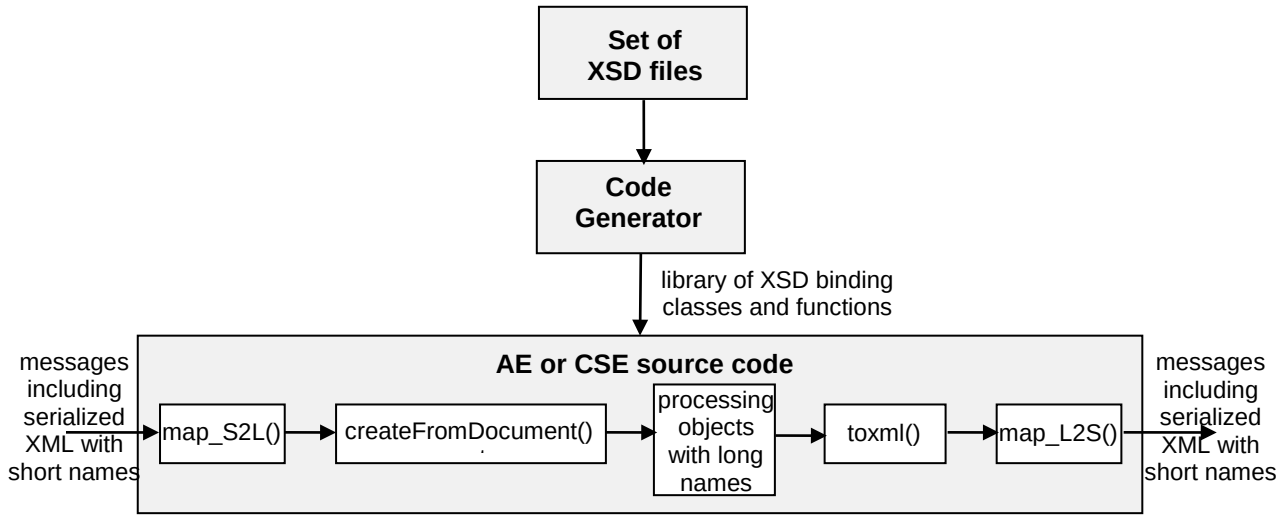

Figure I.2-1: Example AE or CSE implementation: processing based on long names, XML representations using short names

I.3 Example AE/CSE implementation featuring mapping between short and long names for JSON serialization

Figure I.3-1 shows an example implementation which employs JSON serialization. The core of this example implementation is identical with the one described above for XML serialization. In the example it is assumed that for producing a JSON representation which is valid against its associated XSD, an XML file is generated first by means of the toxml() function described in clause I.2 above. In this case the mapping from long to short names can be accomplished also with the map_L2S() function used in the XML serialization example. This XML file can then be converted into a structured data representation that allows direct conversion into JSON. When using Python programming language, the most suitable representation is the dictionary format. In Figure I.3-1, the function denoted as xml2dict(), generates a Python dictionary object which in the final operation step is serialized into the XSD-valid JSON representation by means of the json.dumps() function. In order to comply with the requirements for the JSON representation as defined in clause 8.4, it is necessary to adjust the data type of numeric and list-type elements.

At the receiving side of the described implementation example, received JSON data is converted into a Python dictionary object by means of the json.loads() function. This dictionary object is unparsed by means of a function denoted dict.unparse() in Figure I.3-1 which generates directly an instance of the class applicable to the received data which is defined in SchemaLib. During the unparse operation, the mapping is accomplished between the short names included in the received JSON data object and the long names employed in the class definition included in SchemaLib. The unparse operation also implements validation of the compliance of the received JSON data with the XSD.

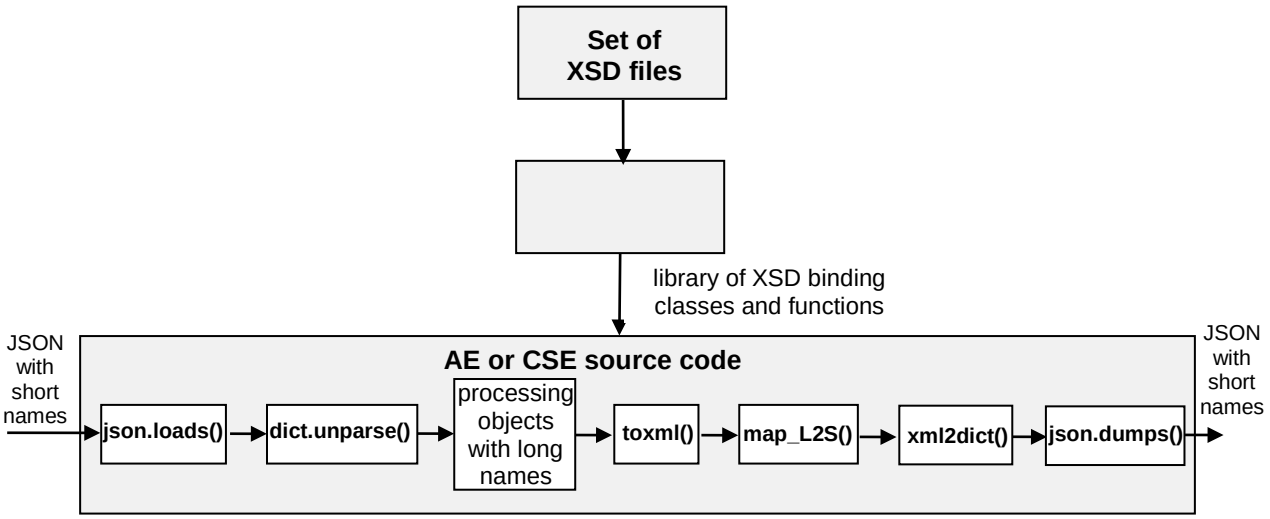

Figure I.3-1: Example AE or CSE implementation with processing based on long names

Annex J (normative): Specializations of <flexContainer> resource

J.1 Introduction

This annex defines each specialization of <flexContainer> resource that are used for generic interworking [36] and AllJoyn interworking [37]. The <flexContainer> resource and procedures are defined in the clause 7.4.37. Since the specialization resources handling procedures are the same as <flexContainer> resource, this annex does not specify them. Also, since all the specialization inherits the universal/common attributes of <flexContainer> resource, this annex does not specify that information.

J.2 Resource type [genericInterworkingService]

This resource type is used for grouping Input and/or Output Datapoints and/or OperationInstances of a Service in the context of Ontology based Interworking. The detailed description of the [genericInterworkingService] resource can be found in clause 9.2 of oneM2M TS-0012 [36].

Table J.2-1: Data type definition of [genericInterworkingService] resource

Data Type ID	File Name	Note
genericInterworkingService	CDT-genericInterworkingService-v3_11_0.xsd	XSD schema for genericInterworkingService resource

Table J.2-2: Resource Specific Attributes of [genericInterworkingService] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
serviceName	M	NP	xs:string	No default.
inputDataPointLinks	O	O	m2m:listOfDataLinks	No default
outputDataPointLinks	O	O	m2m:listOfDataLinks	No default

J.3 Resource type [genericInterworkingOperationInstance]

This resource type and is used for grouping (persistent) Input and/or Output Datapoints and/or (transient) OperationInput / Output of an Operation in the context of Ontology based Interworking. Resources of resource type genericInterworkingOperationInstance are created as child-resources of a genericInterworkingService. The detailed description of the [genericInterworkingService] resource can be found in clause 9.2 of oneM2M TS-0012 [36].

Table J.3-1: Data type definition of [genericInterworkingOperationInstance] resource

Data Type ID	File Name	Note
genericInterworkingOperationInstance	CDT-genericInterworkingOperationInstance-v3_11_0.xsd	XSD schema for genericInterworkingOperationInstance resource

Table J.3-2: Resource Specific Attributes of [genericInterworkingOperationInstance] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
operationName	M	NP	xs:string	No default
inputDataPointLinks	O	O	m2m:listOfDataLinks	No default
outputDataPointLinks	O	O	m2m:listOfDataLinks	No default
inputLinks	O	O	m2m:listOfDataLinks	No default
outputLinks	O	O	m2m:listOfDataLinks	No default
operationState	M	O	xs:string	No default

J.4 Resource type [svcObjWrapper]

This specialization of <*flexContainer*> is intended to be used to wrap a group of child resources related to AllJoyn service objects with minimal overhead. No custom attributes are needed for this specialization.

Table J.4-1: Data type definition of [svcObjWrapper] resource

Data Type ID	File Name	Note
svcObjWrapper	CDT-svcObjWrapper-v3_11_0.xsd	XSD schema for [svcObjWrapper] resource

Table J.4-2: Resource Specific Attributes of [svcObjWrapper] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default

Table J.4-3: Child Resources of [svcObjWrapper] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n
[allJoynApp]	[variable]	0..n

J.5 Resource type [svcFwWrapper]

This specialization of <*flexContainer*> is intended to be used to wrap a group of child resources related to AllJoyn framework services with minimal overhead. No custom attributes are needed for this specialization.

Table J.5-1: Data type definition of [svcFwWrapper] resource

Data Type ID	File Name	Note
svcFwWrapper	CDT-svcFwWrapper-v3_11_0.xsd	XSD schema for [svcFwWrapper] resource

Table J.5-2: Resource Specific Attributes of [svcFwWrapper] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default

Table J.5-3: Child Resources of [svcFwWrapper] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n

J.6 Resource type [allJoynApp]

This specialization of <*flexContainer*> is used to represent a specific instance of an AllJoyn application. This resource shall include the *direction* custom attribute.

Table J.6-1: Data type definition of [allJoynApp] resource

Data Type ID	File Name	Note
allJoynApp	CDT-allJoynApp-v3_11_0.xsd	XSD schema for [allJoynApp] resource

Table J.6-2: Resource Specific Attributes of [allJoynApp] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
direction	M	NP	m2m:allJoynDirection	No default

Table J.6-3: Child Resources of [allJoynApp] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n
[allJoynSvcObject]	[variable]	0..n

J.7 Resource type [allJoynSvcObject]

This specialization of <*flexContainer*> is used to represent a specific instance of an AllJoyn application. This resource shall include the *objectPath* and *enable* custom attributes.

Table J.7-1: Data type definition of [allJoynSvcObject] resource

Data Type ID	File Name	Note
allJoynSvcObject	CDT-allJoynSvcObject-v3_11_0.xsd	XSD schema for [allJoynSvcObject] resource

Table J.7-2: Resource Specific Attributes of [allJoynSvcObject] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
objectPath	M	NP	xs:string	No default.
enable	M	O	xs:boolean	false

Table J.7-3: Child Resources of [allJoynSvcObject] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n
[allJoynInterface]	[variable]	0..n

J.8 Resource type [allJoynInterface]

This specialization of <*flexContainer*> is used to represent a specific implementation of an AllJoyn interface residing in an AllJoyn service object. This resource shall include the *interfaceIntrospectXmlRef* custom attribute.

Table J.8-1: Data type definition of [allJoynInterface] resource

Data Type ID	File Name	Note
allJoynInterface	CDT-allJoynInterface-v3_11_0.xsd	XSD schema for [allJoynInterface] resource

Table J.8-2: Resource Specific Attributes of [allJoynInterface] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
interfaceIntrospectXmlRef	M	NP	xs:anyURI	No default

Table J.8-3: Child Resources of [allJoynInterface] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n
[allJoynInterface]	[variable]	0..n
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n

J.9 Resource type [allJoynMethod]

This specialization of <*flexContainer*> is used to represent a specific method of an AllJoyn interface residing in an AllJoyn service object. No custom attributes are needed for this specialization.

Table J.9-1: Data type definition of [allJoynMethod] resource

Data Type ID	File Name	Note
allJoynMethod	CDT-allJoynMethod-v3_11_0.xsd	XSD schema for [allJoynMethod] resource

Table J.9-2: Resource Specific Attributes of [allJoynMethod] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default

Table J.9-3: Child Resources of [allJoynMethod] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n
[allJoynMethodCall]	[variable]	0..n

J.10 Resource type [allJoynMethodCall]

This specialization of <*flexContainer*> is used to represent a specific calling instance of a method of an AllJoyn interface residing in an AllJoyn service object. This resource shall include the *input, output* and *callStatus* custom attributes.

Table J.10-1: Data type definition of [allJoynMethodCall] resource

Data Type ID	File Name	Note
allJoynMethodCall	CDT-allJoynMethodCall-v3_11_0.xsd	XSD schema for [allJoynMethodCall] resource

Table J.10-2: Resource Specific Attributes of [allJoynMethodCall] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
input	O	NP	xs:string	No default.
callStatus	O	O	xs:string	No default
output	O	O	xs:string	No default

Table J.10-3: Child Resources of [allJoynMethodCall] resource

Child Resource Type	Child Resource Name	Multiplicity
<subscription>	[variable]	0..n
<semanticDescriptor>	[variable]	0..n

J.11 Resource type [allJoynProperty]

This specialization of <*flexContainer*> is used to represent a specific property of an AllJoyn interface residing in an AllJoyn service object. This resource shall include the *currentValue* and *requestedValue* custom attributes.

Table J.11-1: Data type definition of [allJoynProperty] resource

Data Type ID	File Name	Note
allJoynProperty	CDT-allJoynProperty-v3_11_0.xsd	XSD schema for [allJoynProperty] resource

Table J.11-2: Resource Specific Attributes of [allJoynProperty] resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
containerDefinition	M	NP	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default
currentValue	M	O	xs:string	No default
requestedValue	O	O	xs:string	No default

Table J.11-3: Child Resources of [allJoynProperty] resource

Child Resource Type	Child Resource Name	Multiplicity
<*subscription*>	[variable]	0..n
<*semanticDescriptor*>	[variable]	0..n

History

Publication history		
V3.11.2	May 2019	Release 3 - Publication