ONEM2M
TECHNICAL SPECIFICATION

Document Number	TS-0004-V1.0.1
Document Name:	Service Layer Core Protocol Specification
Date:	2015-January-30
Abstract:	The present document specifies the communication protocol(s) for oneM2M compliant Systems, M2M Applications, and/or other M2MSystems.The present document also specifies the 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

Contents

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

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

2.1 Normative references

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

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

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

[3] W3C XMLSchemaP2: "W3C Recommendation (2004), XML Schema Part 2:Datatypes Second Edition.".

[4] oneM2M TS-0005 “Management Enablement (OMA)”

[5] oneM2M TS-0006 “Management Enablement (BBF)”

[6] oneM2M TS-0001 "Functional Architecture"

[7] oneM2M TS-0003 “Security Solutions”

[8] IEEE 754-2008: IEEE. IEEE Standard for Floating-Point Arithmetic. 29 August 2008. http://ieeexplore.ieee.org/servlet/opac?punumber=4610933

[9] IETF RFC 3548: "The Base16, Base32, and Base64 Data Encodings". 2003.

[10] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies". 1996.

[11] IETF RFC 3987:" Internationalized Resource Identifiers (IRIs)" . January 2005.

[12] IETF BCP 47: "Best Current Practices 47". Concatenation of RFC 4646:" Tags for Identifying Languages"(2006) and RFC 4647: "Matching of Language Tags"(2006).

[13] IETF RFC 3588: "Diameter Base Protocol". September 2003.

[14] IETF RFC 6733: "Diameter Base Protocol". October 2012.

[15] 3GPP TS 23.682: "Architecture enhancements to facilitate communications with packet data networks and applications" Release 11.

[16] 3GPP TS 29.368: "Tsp interface protocol between the MTC Interworking Function (MTC-IWF) and Service Capability Server (SCS)" Release 11.

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

[18] IETF RFC 4282: "The Network Access Identifier".

[19] IETF RFC 7159: "The JavaScript Object Notation (JSON) Data Interchange Format".

[20] IETF RFC 4234: "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 6837: " Media Type Specifications and Registration Procedures".

[27] ISO 3601:2004; "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[28] OMA-TS-REST-NetAPI_TerminalLocation-V1_0-20130924-A: "RESTful Network API for Terminal Location", Version 1.0.

2.2 Informative references

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

[i.1] oneM2M Drafting Rules.

NOTE: Available at http://member.onem2m.org/Static_pages/Others/Rules_Pages/oneM2M-Drafting-Rules-V1_0.doc.

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

[i.3] "RESTful Network API for Notification Channel", Open Mobile Alliance™, OMA-TS-REST_NetAPI_NotificationChannel-V1_0.

[i.4] OMA-TS-MLP-V3_4-20130226-C: "Mobile Location Protocol", Version 3.4.

3 Definitions and abbreviations

3.1 Definitions

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

Complex Data Types: is a data type that has a child element.

Enumeration Type: is a data type that enables for a variable to be 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: is a server offering location capabilities.

M2M Area Network: a network provides 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.

Originator: For single-hop case, the Originator is the entity that sends a Request. For multi-hop case, the Originator is the entity that sends the first Request in a sequence of requests.

NOTE: An Originator can either be an AE or a CSE.

Receiver: is the entity that receives the Request.

Receiver CSE: is any CSE that receives a request.

Registrar CSE: CSE is the CSE where an Application or another CSE has registered.

Registree/Registrar CSE: is the CSE that registers with another CSE.

Request: is the message sent from the Originator to the Receiver.

Response: is the message replied to the Request from the Receiver to the Originator.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations and those given in TS-0011 Common Terminology [25] 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 Retrive Update Delete
CRUD+N	Create Retrive Update Delete Notification
CSE-ID	Common Service Entity Identifier
CUDN	Create Update Delete NotifyDAA 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
MQTT	Message Queue Telemetry Transport
MTC-IWF	MachinetType Communications - InterWorking Function
NP	Not Present
OA	Optional Announced
OMA-DM	Open Mobile Alliance Device Management
RD	Retrieve DeleteRFC Request For Comment
RPC	Remote Procedure Call
RSC	Response Status Codes
RUD	Retrieve Update Delete
SCS	Services Capability Server
SP	Service Provider
SP-ID	Service Provider Identifier
TBD	To Be Determined
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UTF	UCS Transformation Format
UUID	Universally Unique Identifier
XML	eXtensible Markup Language
XSD	XML Schema Definition
WLAN	Wireless Local Area Network

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

The following clauses contain the design principles and requirements for the oneM2M protocol.

5.1 Introduction

The oneM2M architecture is resource-based (oneM2M TS-0001 Functional Architecture [6]). The functionality of the system is exposed by means of APIs over all reference points specified in oneM2M TS-0001 Functional Architecture [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 API operations could be fulfilled with the considerations in terms of scalability, extensibility, fault tolerance and robustness, energy efficiency, and self-operation.

Each resource operation comprises a pair of primitives: Request and Response.

In order to provide a well-defined interface for the reference points in oneM2M TS-0001 [6] Functional Architecture, 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 current document provides:

- Protocol design principles and requirements
- Data type definitions;
- Primitive definitions; 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].

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 Functional Architecture [6].

5.1.1 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. The following services are provided by the underlying networks:

- Device triggering (see Annex B)
- Location request (see Annex G)
- Device Management (see clause 7.2.4)

5.2 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 TS-0001 Functional Architecture [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 TS-0001 Functional Architecture [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) Resource has a representation (see [i.2]) that shall be transferred and manipulated with the verbs. These verbs are identified as operations in TS-0001 Functional Architecture [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 TS-0001 Functional Architecture [6], clause 10.2.

7) Primitives shall include attributes in accordance with TS-0001 Functional Architecture [6] for a specific resource.

8) Primitive shall be self-descriptive and contain all the information needed for the receiver of the primitives to handle the primitives.

9) Primitive should be idempotent operations which mean no matter how many times the primitive is sent, the result doesn't change, in accordance to [i.2].

10) Primitives shall be mapped on the transport layer protocols.

5.3 Primitives

5.3.1 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 5.3.1-1, the primitives are mapped to application layer communication protocols such as HTTP, CoAP or MQTT which use TCP or UDP on the transport layer. The specification of primitives, however, is independent of underlying communication protocols and allow introduction of bindings to other communication protocols.

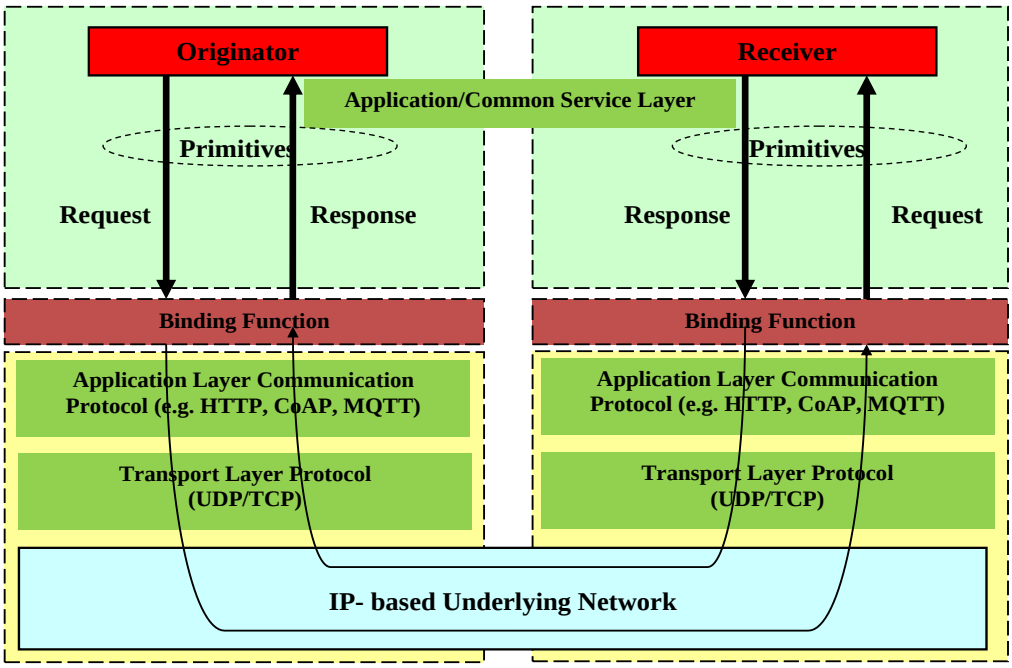

Figure 5.3.1-1: Communication model using Request and Response primitives over an IP-based Underlying Network

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.

5.3.2 Primitives modelling

Primitives are modelled as follows.

A primitive is represented in 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.

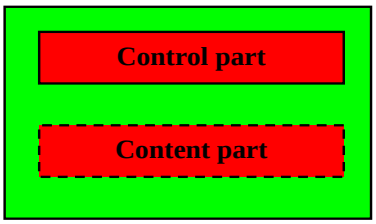

Figure 5.3.2-1: Primitives modelling

5.3.3 Primitive principles

Execution of one primitive shall finish completely before execution of a subsequent primitives 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.

5.3.4. Serialization of primitives

When transferred over a oneM2M reference point while using a communication protocol such as HTTP, CoAP or MQTT, 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 or XML documents used by oneM2M protocol bindings.

5.4 Design principles

5.4.1 Introduction

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

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

5.4.2 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.

5.4.3 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.

5.4.4 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.

5.4.5 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.

5.4.6 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.

5.4.7 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 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 TS-0008, TS-0009 and TS-0010.

Any data types of XML elements defined 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.

The XSD filesreferenced in the present document shall serve following purposes:

1) Provide an unambiguous definition of XML element names and data types used for
   a) resource representations,
   b) resource attributes,
   c) Request and Response primitives (including Notification primitive),
   d) parameters used in Request and Response primitives.
2) Help to identify and avoid that equivalent data types are defined multiple times with different names.
3) Provide a testable definition of the value range of data elements (e.g. allowed number range, allowed characters or character patterns, allowed enumeration values).
4) Provide a testable definition of the presence of mandatory elements (“minOccurs=1”) and of cardinality not larger than a given limit (e.g. “maxOccurs=1”) in XML representations of data objects (i.e. resource instantiations and primitive parameters).
5) Provide a testable definition of the correct sequence of occurrence of each element of a data object (where correct sequence is required).

6) Enable the use of development tools that generate executable code for data object processing from the XSD.

7) 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 this specification 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: M2M implementations are required to validate the data received in incoming primitives in accordance with this specification, but this specification 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

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 Functional Architecture [6].

6.2.1 Summary of oneM2M Identifiers

shows the summary of M2M Identifers defined in oneM2M TS-0001 Functional Architecture [6].

Table 6.2.1-1: M2M Identifiers

Identifier	Data Type	Description
M2M-SP-ID	m2m:ID (see clause 6.3.2)	A globally unique ID as specified in [6]
App-ID	m2m:ID (see clause 6.3.2)	The identifier is specified in[6]
AE-ID	m2m:ID (see clause 6.3.2)	A globally unique ID as specified in [6]
CSE-ID	m2m:ID (see clause 6.3.2)	A globally unique ID as specified in [6]
M2M-Node-ID	m2m:nodeID (see clause 6.3.2)	A globally unique ID as specified in[6]
M2M-Sub-ID	m2m:ID (see clause 6.3.2)	A globally unique ID as specified in [6]
M2M-Request-ID	m2m:requestID (see clause 6.3.2)	A unique ID as specified in [6]
M2M-Ext-ID	m2m:externalID (see clause 6.3.4)	The identifier is specified in [6]
UNetwork-ID	m2m:ID (see clause 6.3.2)	A unique ID as specified in [6]
Trigger-Recipient-ID	xs:unsignedInt	The identifier is specified in [6]

6.2.2 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 specification assumes each oneM2M entity has the CSE-PoA address of its Registrar CSE in advance.

When the oneM2M entity is communicating to another oneM2M entity , the address appearing in the oneM2M primitive (e.g. at From or To parameter) shall be the absolute form of AE-ID or CSE-ID defined in oneM2M TS-0001 [6].

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

```
absolute-CSE-ID = "//" M2M-SP-ID  SP-"/" Relative-CSE-ID
```

EXAMPLE 1 Starts:

EXAMPLE: `//myoperator.com/cse1`

This is example of CSE-ID, “myoperator.com” is the M2M-SP-ID and “cse1” is M2M-SP relative CSE-ID.

EXAMPLE 1 Ends:

The AE-ID which is assigned by M2M Service Provider (SP relative AE-ID), or by Registrar CSE locally (CSE relative AE-ID).

The syntax of AE-ID in ABNF notion [20] is following:

```
AE-ID = SP-relative-AE-ID / CSE-relative-AE-ID

absoluteSP-relative-AE-ID = "//" M2M-SP-ID "/" S-AE-ID-Stem

SPCSE-relative-AE-ID = CSE-ID "/" C-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

unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
```

EXAMPLE 2 Starts:

EXAMPLE: `//myoperator.com/S563423`

This is example of absolute-SP-relative AE-ID.

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

This is example of 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 2 Ends:

6.2.3 oneM2M Resource Addressing

The authorized oneM2M entities can operate oneM2M Resourrce by specifying the Resource Identifier as the target address.

There are 2 forms of oneM2M Resource Identifer is defined in clause 7.2 of the oneM2M TS-0001[6].

1) Hierachical Address: the address which is constructed as relative path from the CSEBase resource via parent resources.

2) Non-hierarchical Address: the address which uniquely identifies the resource in the M2M-SP's domain.

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.

The address appearing in 'ChildResourceRef' may be used as in M2M-SP relative form, if it is obvious which M2M-SP or CSE-ID should be added.

The syntax of the oneM2M Resource Identifier (resource-ID) in ABNF notion [20] is following:

```
resource-address = (hierarchical-resouce-address / non-hierachical-resource-address) [ "#" targeted-
     attribute-name ]

hierachical-resource-address = [[SP-ID] CSE-ID] 1*("/" resource-name)

non-hierarchical-resource-address = [[SP-ID] CSE-ID] non-hierarchical-resource-ID

resource-name = 1*unreserved
```

6.3 Common data types

The following sub-clauses define the data format of resource attributes and parameters used in primitives.

6.3.1 Simple data types incorporated from XML schema

The following 'built-in data types' 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 6.3.1-1: 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:dateTime	dateTime represents instants of time, optionally marked with a particular time zone offset. Values representing the same instant but having different time zone offsets are equal but not identical.	
xs:time	time represents instants of time that recur at the same point in each calendar day, or that occur in some arbitrary calendar day.	
xs:date	date represents top-open intervals of exactly one day in length on the timelines of dateTime, beginning on the beginning moment of each day, up to but not including the beginning moment of the next day). For non-time zoned values, the top-open intervals disjointly cover the non-time zoned timeline, one per day. For time zoned values, the intervals begin at every minute and therefore overlap.	
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 RFC 3548 [9], which is derived from the encoding described in 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 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	

Data Type	Description	Notes
	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.	
xs:language	language represents formal natural language identifiers, as defined by BCP 47[12].	
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.	

6.3.2 oneM2M simple data types

Table 6.3.2-1 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-v1_0_0.xsd.

The types in table 6.3.2-1 are either:

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

The oneM2M-defined enumeration data types are defined in clause 6.3.3.

Table 6.3.2-1: oneM2M Simple Data Types

XSD type name	Type Name	Examples	Description
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		
m2m:externalID	M2M-EXT-ID	urn:gsma:imei:90420156-025763-0;vers=0	The identifier of the node for the underlying network provider. In 3GPP case, the accessID is mapped to External Identifier as specified in TS 23.003 [17]. Or (MSISDN) The identifier of the node as specified in TS 23.003 [17],
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 Types	//IN-CSEID.m2m.myoperator.org/93405	Used to represent an AccessControlPolicy identifier. This can be either a URI or an opaque token
m2m:labelsType	Labels	(printers networkwifi1 vender1)	A list of tokens used as keys for discovering resources (searching wifi connected printer from vender 1)
m2m:networkaccessID	Network Access Identifier	user@realm	The networkaccessIdentifier is a standard way of identifying users who request access to a network as specified at IETF RFC 4282 [18].
m2m:listOfM2MID	List of M2M identifiers		xs:list of elements of data type m2m:ID
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:backOffParameters	List of Backoff Parameters	(100 100 2000)	Ordered sequence of 3 values of data type xs:nonNegativeInteger representing backoffTime, backoffTimeIncrement, maximumBackoffTime

			(in units of milliseconds)
m2m:ipv4	IPv4 address string with optional CIDR suffix	10.125.0.0/16, 122.77.12.1	Required in m2m:acr
m2m:ipv6	IPv6 address string with optional CIDR suffix	::/0, Fadf:ddd0::/32, abcd:ffff:abb0:aaaa::/64	Required in m2m:acr
m2m:countryCode	Country Code	KR	2-character country code as defined by ISO-3166
m2m:poaList	List of PointOfAccess strings	http://172.25.0.10:8080, coap://m2m.sp.com	list of xs:string. Each pointOfAccess entry in list is represented as a string containing the underlying transport protocol as well as the IP address and port (or an FQDN).
m2m:timestamp	Time stamp string	20141003T112032	DateTime string of 'Basic Format' specified in ISO8601 [27]. Time zone shall be interpreted as UTC timezone.
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:contentInfo	Content Information	application/xml:2	A string consisting of a media type optionally followed by a m2m:encoding separated by ':' character. If the encoding portion is omitted, value 0 (plain) shall be applied ':'..
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
m2m:listOfEventCatWithDef	List of m2m:eventCatWithDef	0 1 101	
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 7.3.9.1

6.3.3 oneM2M enumerated data types

6.3.3.1 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 6.3.3.2. Table 6.3.3.1-1 shows the example of Enumeration Type definition for m2m:enumFooType.

Table 6.3.3.1-1: 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-v1_0_0-<<date of publication>>.xsd. Figure 6.3.3.1-1 shows the example of 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 6.3.3.1-1: Example of XSD version of oneM2M Enumeration Type

6.3.3.2 Enumeration type definitions

6.3.3.2.1 m2m:resourceType

Table 6.3.3.2.1-1: 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	
19	serviceSubscribedAppRule	
20	serviceSubscribedNode	
21	statsCollect	
22	statsConfig	
23	subscription	
10001	accessControlPolicyAnnc	
10002	AEAnnc	
10003	containerAnnc	
10004	contentInstanceAnnc	
10009	groupAnnc	
10010	locationPolicyAnnc	
10013	mgmtObjAnnc	
10014	nodeAnnc	
10016	remoteCSEAnnc	
10018	scheduleAnnc	
NOTE: See clause 6.4.1 “Request message parameter data types”		

6.3.3.2.2 m2m:cseTypeID

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

Table 6.3.3.2.2-1: Interpretation of cseTypeID

Value	Interpretation	Note
1	IN_CSE	
2	MN_CSE	
3	ASN_CSE	
NOTE: See clause 7.3.4 “Resource Type remoteCSE”		

6.3.3.2.3 m2m:locationSource

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

Table 6.3.3.2.3-1: Interpretation of locationSource

Value	Interpretation	Note
1	Network_based	
2	Device_based	
3	Sharing_based	
NOTE: See clause 7.3.10 “Resource Type locationPolicy”		

6.3.3.2.4 m2m:stdEventCats

Used for ***ec*** parameter in request and ***eventCat*** attribute of <delivery> resource and cmdh policy resource types.

Table 6.3.3.2.4-1: Interpretation of stdEventCats

Value	Interpretation	Note
1	Default	
2	Immediate	
3	BestEffort	
4	Latest	
NOTE: See clause 7.3.11 “Resource Type delivery” and Annex D.12 “Resource cmdhPolicy”		

6.3.3.2.5 m2m:operation

Used for ***Operation*** parameter in request and ***operation*** attribute in <request> resource as well as operationMonitor.

Table 6.3.3.2.5-1: Interpretation of operation

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

6.3.3.2.6 m2m:responseType

Used for ***Response Type*** parameter (as a part of responseTypeInfo, See Clause 6.3.4.29) in request .

Table 6.3.3.2.6-1: Interpretation of responseType

Value	Interpretation	Note
1	nonBlockingRequestSynch	
2	nonBlockingRequestAsynch	
3	blockingRequest	
NOTE: See clause 6.4.1 “Request message parameter data types”		

6.3.3.2.7 m2m:resultContent

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

Table 6.3.3.2.7-1: 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	
NOTE: See clause 6.4.1 “Request message parameter data types”		

6.3.3.2.8 m2m:discResType

Used in ***metaInformation*** attribute in <request> resource

Table 6.3.3.2.8-1: Interpretation of discResType

Value	Interpretation	Note
1	hierarchical	
2	non_hierarchical	
3	cseID and resourceID	
NOTE: See clause 6.4.1 “Request message parameter data types”		

6.3.3.2.9 m2m:responseStatusCode

See clause 6.6.3 "Current Response Status Codes"

Table 6.3.3.2.9-1: Interpretation of responseStatusCode

Value	Interpretation	Note
(‘Numeric Code’ in Clause 6.6.3)	(‘Description’ in clause 6.6.3)	

6.3.3.2.10 m2m:requestStatus

Used for ***requestStatus*** attribute in <request> resource.

Table 6.3.3.2.10-1: Interpretation of requestStatus

Value	Interpretation	Note
1	COMPLETED	
2	FAILED	
3	PENDING	
4	FORWARDED	
NOTE: See clause 7.3.12 “Resource Type request”		

6.3.3.2.11 m2m:memberType

Used for ***memberType*** attribute in <member> resource.

Table 6.3.3.2.11-1: Interpretation of memberType

Value	Interpretation	Note
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	mixed	
NOTE: See clause 7.3.13 “Resource Type group”		

6.3.3.2.12 m2m:consistencyStrategy

Used for ***consistencyStrategy*** attribute in <group> resource.

Table 6.3.3.2.12-1: Interpretation of consistencyStrategy

Value	Interpretation	Note
1	ABANDON_MEMBER	
2	ABANDON_GROUP	
3	SET_MIXED	
NOTE: See clause 7.3.13 “Resource Type group”		

6.3.3.2.13 m2m:cmdType

Used for ***cmdType*** attribute in <mgmtCmd> resource.

Table 6.3.3.2.13-1: Interpretation of cmdType

Value	Interpretation	Note
1	RESET	
2	REBOOT	
3	UPLOAD	
4	DOWNLOAD	
5	SOFTWAREINSTALL	
6	SOFTWAREUNINSTALL	
7	SOFTWAREUPDATE	
NOTE: See clause 7.3.16 “Resource Type mgmtCmd”		

6.3.3.2.14 m2m:execModeType

Used for ***execModeType*** attribute in <mgmtCmd> and <execInstance> resource.

Table 6.3.3.2.14-1: Interpretation of execModetType

Value	Interpretation	Note
1	IMMEDIATEONCE	
2	IMMEDIATEREPEAT	
3	RANDOMONCE	
4	RANDOMREPEAT	
NOTE: See clause 7.3.16 “Resource Type mgmtCmd” and Clause 7.3.17 “Resource Type execInstance”		

6.3.3.2.15 m2m:execStatusType

Used for ***execStatusType*** attribute in <execInstance> resource.

Table 6.3.3.2.15-1: Interpretation of execStatusType

Value	Interpretation	Note
1	INITIATED	
2	PENDING	
3	FINISHED	
4	CANCELLING	
5	CANCELLED	
6	STATUS_NON_CANCELLABLE	
NOTE: See clause 7.3.17 “Resource Type execInstance”		

6.3.3.2.16 m2m:execResultType

Used for execStatusType attribute in <execInstance> resource.

Table 6.3.3.2.16-1: Interpretation of execResultType

Value	Interpretation	Note
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	
17	STATUS_FILE_TRANSFER_FAILED	
18	STATUS_FILE_TRANSFER_SERVER_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	
NOTE: See clause 7.3.16 “Resource Type mgmtCmd”		

6.3.3.2.17 m2m:pendingNotification

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

Table 6.3.3.2.17-1: Interpretation of pendingNotification

Value	Interpretation	Note
1	sendLatest	
2	sendAllPending	
NOTE: See clause 7.3.8 “Resource Type subscription”		

6.3.3.2.18 m2m:notificationContentType

Table 6.3.3.2.18-1: Interpretation of notificationContentType

Value	Interpretation	Note
1	modifiedAttributes	
2	wholeResource	
3	referenceOnly	
NOTE: See clause 7.3.8 “Resource Type subscription”		

6.3.3.2.19 m2m:resourceStatus

This is used for eventNotificationCriteria.

Table 6.3.3.2.19-1: Interpretation of resourceStatus

Value	Interpretation	Note
1	childCreated	
2	childDeleted	
3	updated	
4	deleted	
NOTE: See clause 7.4.1.1 “Definition of Notification”		

6.3.3.2.20 m2m:status

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

Table 6.3.3.2.20-1: Interpretation of status

Value	Interpretation	Note
1	Successful	
2	Failure	
3	In_Process	
NOTE: See clause D.2, D.3 firmware and software management		

6.3.3.2.21 m2m:batteryStatus

This is used for [battery] resource.

Table 6.3.3.2.21-1: 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 annex D.7 battery management		

6.3.3.2.22 m2m:mgmtDefinition

This is used for <mgmtObj> resource.

Table 6.3.3.2.22-1: 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	
0	Unspecified	Permits vendor-specific extensions
NOTE: See clause 7.3.15 mgmtObj		

6.3.3.2.23 m2m:logTypeId

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

Table 6.3.3.2.23-1: Interpretation of logTypeId

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

6.3.3.2.24 m2m:logStatus

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

Table 6.3.3.2.24-1: 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.

6.3.3.2.25 m2m:eventType

Used for ***eventType*** attribute in <eventConfig> resource.

Table 6.3.3.2.25-1: Interpretation of eventType

Value	Interpretation	Note
1	DATAOPERATION	
2	STORAGEBASED	
3	TIMERBASED	
NOTE: See clause 7.3.24 “Resource Type eventConfig”		

6.3.3.2.26 m2m:statsRuleStatusType

Used for ***statsRuleStatusType*** attribute in <statsCollect> resource.

Table 6.3.3.2.26-1: Interpretation of statsRuleStatusType

Value	Interpretation	Note
1	ACTIVE	
2	INACTIVE	
NOTE: See clause 7.3.25 “Resource Type statsCollect”		

6.3.3.2.27 m2m:statModelType

Used for ***statModelType*** attribute in <statsCollect> resource.

Table 6.3.3.2.27-1: Interpretation of statModelType

Value	Interpretation	Note
1	EVENTBASED	
NOTE: See clause 7.3.25 “Resource Type statsCollect”		

6.3.3.2.28 m2m:encodingType

Used for describing encoding type which is applied on the ***content*** attribute of the ***contentInstance*** resource.

Table 6.3.3.2.28-1: 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	

6.3.3.2.29 m2m:accessControlOperations

Used for accessControlPolicys.

Table 6.3.3.2.29-1: Interpretation of accessControlOperations

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

6.3.3.2.30 m2m:SRole-ID

Used for <m2mServiceSubscriptionProfile>

Table 6.3.3.2.30-1: Interpretation of SRole-ID

Value	Interpretation	Note
"01-001"	Software Management	
"02-001"	Device Configuration	
"02-002"	Device Diagnostics and Management	
"02-003"	Device Firmware Management	
"02-004"	Device Topology	
"03-001"	Location	
"04-001"	Basic Data	
"05-001"	Onboarding	
"06-001"	Security Administration	
"07-001"	Groups Management	
"08-001"	Event Collection	
NOTE: This is an enumeration of String values		

6.3.4 Complex data types

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-v1_0_0.xsd.

In addition, each oneM2M resource has a corresponding complex data type. These are described in Clause 6.5.

6.3.4.1 m2m:deliveryMetaData

Used for ***deliveryMetaData*** attribute in <delivery> resource.

Table 6.3.4.1-1: Type Definition of m2m:deliveryMetadata

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

6.3.4.2 m2m:aggregatedRequest

Used for ***aggregatedRequest*** attribute in <delivery> resource.

Table 6.3.4.2-1: Type Definition of m2m:aggregatedRequest

Element Path	Element Data Type	Multiplicity	Note
request	(anonymous)	1..n	
request/operation	m2m:operation	1	See Clause 6.3.3.2.5
request/to	xs:anyURI	1	
request/from	m2m:ID	1	See Clause 6.3.2
request/requestIdentifier	m2m:requestID	1	See Clause 6.3.2
request/content	m2m:primitiveContent	0..1	See Clause 6.3.4.4
request/metaInformation	m2m:metaInformation	0..1	See Clause 6.3.4.3

6.3.4.3 m2m:metaInformation

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

Table 6.3.4.3-1: Type Definition of m2m:metaInformation

Element Path	Element Data Type	Multiplicity	Note
resourceType	m2m:resourceType	0..1	See Clause 6.3.3.2.1
name	xs:string	0..1	
originatingTimestamp	m2m:timestamp	0..1	
requestExpirationTimestamp	m2m:timestamp	0..1	
resultExpirationTimestamp	m2m:timestamp	0..1	
operationExecutionTime	m2m:timestamp	0..1	
responseType	m2m:responseType	0..1	See Clause 6.3.3.2.6
resultPersistence	m2m:timestamp	0..1	
resultContent	m2m:resultContent	0..1	See Clause 6.3.3.2.7
eventCategory	m2m:eventCat	0..1	See Clause 6.3.2
deiveryAggregation	xs:boolean	0..1	
groupRequestIdentifier	xs:string	0..1	
filterCriteria	m2m:filterCriteria	0..1	See Clause 6.3.4.7
discoveryResultType	m2m:discResType	0..1	See Clause 6.3.3.2.8

6.3.4.4 m2m:primitiveContent

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

See clause 7.1.1.1 and 7.1.1.2 .

6.3.4.5 m2m:batchNotify

Used for ***batchNotify*** attribute in <subscription> resource.

Table 6.3.4.5-1: Type Definition of m2m:batchNotify

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

6.3.4.6 m2m:eventNotificationCriteria

Used for ***eventNotificationCriteria*** of a <subscription> resource.

Table 6.3.4.6-1: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	
resourceStatus	m2m:resourceStatus	0..n	
operationMonitor	m2m:operation	0..5	
attribute	m2m:attribute	0..n	

6.3.4.7 m2m:filterCriteria

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

Table 6.3.4.7-1: Type Definition of m2m:filterCriteria

Element Path	Element Data Type	Mutiplicity	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	
resourceType	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	
filterUsage	m2m:filterUsage	0..1	
limit	xs:nonNegativeInteger	0..1	

6.3.4.8 m2m:attribute

Used in m2m:eventNotificationCriteria and m2m:filterCriteria.

Table 6.3.4.8-1: Type Definition of m2m:attribute

Element Path	Element Data Type	Multiplicity	Note
@name	xs:NCName	1	
(base content)	xs:anyType	1	

6.3.4.9 m2m:attributeList

Used in the Content parameter of a Primitive

Table 6.3.4.9-1: Type Definition of m2m:attributeList

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

6.3.4.10 m2m:scheduleEntries

Table 6.3.4.10-1: Type Definition of m2m:scheduleEntries

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

6.3.4.11 m2m:aggregatedNotification

Used in the Notification Data Object.

Table 6.3.4.11-1: Type Definition of m2m:aggregatedNotification

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

6.3.4.12 m2m:notification

Table 6.3.4.12-1: Type Definition of m2m:notification

Element Path	Element Data Type	Multiplicity	Note
notificationEvent	(anonymous)	0..1	
notificationEvent/representation	xs:anyType	0..1	Representation of resource modification in XML/JSON representation.
notificationEvent/resourceStatus	m2m:resourceStatus	0..1	
notificationEvent/operationMonitor	(anonymous)	0..1	
notificationEvent/operationMonitor/operation	m2m:operation	1	m2m:operation
notificationEvent/operationMonitor/originator	m2m:ID	1	m2m:ID
verificationRequest	xs:boolean	0..1	
subscriptionDeletion	xs:boolean	0..1	
subscriptionReference	xs:anyURI	1	
creator	m2m:ID	0..1	
notificationForwardingURI	xs:anyURI	0..1	

6.3.4.13 m2m:actionStatus

Table 6.3.4.13-1: 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 6.3.2.2 1

6.3.4.14 m2m:anyArgType

Table 6.3.4.14-1: Type Definition of m2m:anyArgType

Element Path	Element Data Type	Multiplicity	Note
name	xs:NCName		
type	xs:anyType		

6.3.4.15 m2m:resetArgsType

Table 6.3.4.15-1: Type Definition of m2m:resetArgsType

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

6.3.4.16 m2m:rebootArgsType

Table 6.3.4.16-1: Type Definition of m2m:rebootArgsType

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

6.3.4.17 m2m:uploadArgsTypes

Table 6.3.4.17-1: 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	

6.3.4.18 m2m:downloadArgsType

Table 6.3.4.18-1: 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	

6.3.4.19 m2m:softwareInstallArgsType

Table 6.3.4.19-1: 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	

6.3.4.20 m2m:softwareUpdateArgsType

Table 6.3.4.20-1: 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	

6.3.4.21 m2m:softwareUninstallArgsType

Table 6.3.4.21-1: 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	

6.3.4.22 m2m:execReqArgsListType

Table 6.3.4.22-1: Type Definition of m2m:execReqArgsListType

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

This type is an xs:choice. It shall contain elements from no more than one row listed in the table above.

6.3.4.23 m2m:mgmtLinkRef

Table 6.3.4.23-1: 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	xs:string	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 the above table, 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.

6.3.4.24 m2m:resourceWrapper

Table 6.3.4.24-1: Type Definition of m2m:resourceWrapper

Element Path	Element Data Type	Multiplicity	Note
(base content)	m2m:resource	1	Resource element as described in clause 7.3
@URI	xs:anyURI	1	Hierarchical URI of the resource

In the above table, 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.

6.3.4.25 m2m:setOfAcrs

Table 6.3.4.25-1: Type Definition of m2m:setOfAcrs

Element Path	Element Data Type	Multiplicity	Note
accessControlRules	m2m:accessControlRule	1..n	Data type of privileges and selfPrivileges attributes

6.3.4.26 m2m:accessControlRule

Table 6.3.4.26-1: Type Definition of m2m:accessControlRule

Element Path	Element Data Type	Multiplicity	Note
accesscControlOriginators	list of xs:anyURI	1	
accessControlOperations	m2m:accessControlOperations	1	
accessControlContexts		0..1	
accessControlContexts/accessControlWindow	m2m:scheduleEntry	0..n	
accessControlContexts/accessControlIpAddresses	list of m2m:ipv4 or list of m2m:ipv6	0..1	
accessControlContexts/accessControlLocationRegions	m2m:locationRegion	0..n	

6.3.4.27 m2m:locationRegion

Table 6.3.4.27-1: 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

6.3.4.28 m2m:childResourceRef

Table 6.3.4.28-1: Type Definition of m2m:childResourceRef

Element Path	Element Data Type	Multiplicity	Note
(base content)	xs:anyURI	1	URI of the child resource
@name	xs:string	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*

In the above table, 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.

6.3.4.29 m2m:responseTypeInfo

Table 6.3.4.29-1: Type Definition of m2m:responseTypeInfo

Element Path	Element Data Type	Multiplicity	Note
responseType	m2m:responseType	1	See Clause 6.3.3.2.6
notificationURI	xs:anyURI	0..unbounded	This element may be included only when the responseType is set to "2" (nonBlockingRequest Asynch),

6.3.4.30 m2m:rateLimit

Used in <subscription>.

Table 6.3.4.30-1: Type Definition of m2m:rateLimit

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

6.3.4.31 m2m:operationResult

Used for ***operationResult*** attribute in <request> resource.

Table 6.3.4.31-1: Type Definition of m2m:operationResult

Element Path	Element Data Type	Multiplicity	Note
content	m2m:primitiveContent	0..1	See Clause 6.3.4.4
event Category	m2m:eventCat	0..1	See Clause 6.3.2
from	m2m:ID	0..1	See Clause 6.3.2
originating Timestamp	m2m:timestamp	0..1	
request Identifier	m2m:requestID	1	See Clause 6.3.2
result Expiration Timestamp	m2m:timestamp	0..1	
to	xs:anyURI	0..1	
response Status Code	m2m:responseStatusCode	1	See Clause 6.3.3.2.9

6.3.4.32 m2m:aggregatedResponse

Used when aggregating responses by a group.

Table 6.3.4.32-1: Type Definition of m2m:aggregatedResponse

Element Path	Element Data Type	Multiplicity	Note
responsePrimitive	See Table 6.4.2-1 for detail.	1..n	

6.3.5 Universal and Common attributes

TS-0001 Functional Architecture [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). The type and values shall be supported according to the description given inTable 6.3.5-1.

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 6.3.5-1: 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 hosing CSE and used for non hierarchical addressing method
parentID	m2m:nhURI	This attribute is determined by the hosting CSE and specified in all resource types except of <CSEBase>
creationTime	m2m:timestamp	This attribute is determined by the hosting CSE when the resource is locally created
lastModifiedTime	m2m:timestamp	This attribute is determined by the hosting CSE when the addressed resource is modified by means of the UPDATE operation
labels	list of xs:token	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:token	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	xs:NCName	

Table 6.3.5-2 describes some complex types that group together the universal and common attributes, to be used by Resource Type definitions. Note that ***stateTag*** only appears in four resource types, and so is not included in these definitions, instead it is declared in the XSD files of the resources that need it.

Table 6.3.5-2: Complex Data Types declaring groups of resource common attributes

XSD type name	Child Elements	Child Element Datatype	Multiplicity	Description
m2m:resource	@resourceName	xs:NCName	1	
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	labels	m2m:labelsType	0..1	
m2m:regularResource	@resourceName	xs:NCName	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	
	accessControlPolicyIDs	m2m:acpType	0..1	
	creationTime	m2m:timestamp	1	
	expirationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	stateTag	xs:nonNegativeInteger	1	
	labels	m2m:labelsType	0..1	
	accessControlPolicyIDs	m2m:acpType	0..1	
	expirationTime	m2m:timestamp	1	
m2m:announceableResource	@resourceName	xs:NCName	1	Declares the universal / common attributes included in the majority of announceable resource types.
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	xs:anyURI	1	
	accessControlPolicyIDs	m2m:acpType	0..1	
	creationTime	m2m:timestamp	1	
	expirationTime	m2m:timestamp	1	
	lastModifiedTime	m2m:timestamp	1	
	stateTag	xs:nonNegativeInteger	1	
	Labels	m2m:labelsType	0..1	
	Link	xs:anyURI	0..1	
	announceTo	list of xs:anyURI	0..1	
	announcedAttribute	list of xs:token	0..1	
m2m:announcedResource	@resourceName	xs:NCName	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:labelsType	0..1	
	accessControlPolicyIDs	m2m:acpType	0..1	
	expirationTime	m2m:timestamp	1	
	link	list of xs:token	0..1	
m2m:announceableSubordinateResource	@resourceName	xs:NCName	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:labelsType	0..1	
	expirationTime	m2m:timestamp	1	
	announceTo	list of xs:anyURI	0..1	
	announcedAttribute	list of xs:token	0..1	
m2m:announcedSubordinateResource	@resourceName	xs:NCName	1	Declares the common / universal attributes used in the announced variants of the subordinate
	resourceType	m2m:resourceType	1	
	resourceID	m2m:ID	1	
	parentID	m2m:nhURI	1	
	creationTime	m2m:timestamp	1	

	lastModifiedTime	m2m:timestamp	1	resource types
	labels	m2m:labelsType	0..1	
	expirationTime	m2m:timestamp	1	
	link	list of xs:token	0..1	

NOTE: In the above table, 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.

6.3.6 Filter criteria

The request message parameter Filter Criteria shall be specified as combination of following sub-parameters.

6.3.6.1 creationTime condition

The condition matches with the resources which the *creationTime* attribute is chronologically in specified range.

Table 6.3.6.1-1: Defition of Create Time condition

Format Variants	Definition	Note
createdBefore	m2m:timestamp	
created After	m2m:timestamp	
NOTE: Both createdBefore and createdAfter may be specified same time, but createdAfter shall be the timestamp before createdBefore.		

6.3.6.2 lastModifiedTime condition

The condition matches with the resources which the *lastModifiedTime* attribute is chronologically in specified range.

Table 6.3.6.2-1: Defition of LastModified Time condition

Format Variants	Definition	Note
modifiedSince	m2m:timestamp	
unmodifiedSince	m2m:timestamp	
NOTE: Both modifiedSince and unmodifiedSince may be specified same time, but modifiedSince shall be the timestamp before unmodified Since.		

6.3.6.3 State Tag condition

The condition matches with the <*contentInstance*> resources which the *stateTag* attribute is in specified numeric value range.

Table 6.3.6.3-1 Defition of State Tag condition

Format Variants	Definition	Note
stateTagSmaller	xsd:positiveInterger	
stateTagBigger	xsd:nonNegativeInteger	
NOTE:		

6.3.6.4 expirationTime condition

The condition matches with the *expirationTime* attribute of the resources is chronologically in specified chnological time range.

Table 6.3.6.4-1: Defition of ExirationTime condition

Format Variants	Definition	Note
exprieBefore	m2m:timestamp	
expireAfter	m2m:timestamp	
NOTE: Both expireBefore and expireAfter may be specified same time, but expireBefore shall be the timestamp after expireAfter.		

6.3.6.5 labels Match condition

The condition matches with the resource which *labels* attribute contains the specified value.

Table 6.3.6.5-1: Defition of Lebel Match condition

Format Variants	Definition	Note
label contains	xs:token	
any of label matches	list of xs:token	separator of list elements shall be specified in protocol bindings.
NOTE:		

6.3.6.6 resourceType Match condition

The condition matches with the resource which *resourceType* attribute value is specified value.

Table 6.3.6.6-1: Defition of Resource Type Match condition

Format Variants	Definition	Note
single type	m2m:resourceType	
multiple types	list of m2m:resourceType	separator of list elements shall be specified in protocol bindings.
NOTE:		

6.3.6.7 contentSize condtion

The condition matches with the *contentSize* attribute of the <*contentInstance*> resources is in range of specified numerica value range.

Table 6.3.6.7-1: Defition of Content Size Match condition

Format Variants	Definition	Note
sizeBelow	xs:nonNegativeInteger.	
sizeAbove	xs:nonNegativeInteger.	
NOTE:		

6.3.6.8 typeOfContent condition

The condition matches with the *typeOfContent* attribute of the <*contentInstance*> resource is the specified value.

Table 6.3.6.8-1: Defition of Content Type Match condition

Format Variants	Definition	Note
content type	xs:token	The one of the Content-Type string shall be specified
content sub-types	xs:token after ':' character.	The one of the Content sub-type strings shall be specified.
NOTE:		

6.3.6.9 attribute Match condition

The condition matches with the resources which all attribute/value pairs are specified combination.

Table 6.3.6.9-1: Defition of Attribute Match condition

Format Variants	Definition	Note
single pair	concatenation of xs:token with ':' character.	first token shall be shortname of attribute and second token shall be its value.
multiple pairs	list of single pairs.	separator of list elements shall be specified in protocol bindings.
NOTE:		

6.3.6.10 Limit results request parameter

Limitation the number of matching resources to the specified value.

Table 6.3.6.10-1: Defition of Limit conditions

Format Variants	Definition	Note
limits	xs:nonNegativeInteger	
NOTE:		

6.3.6.11 Filter Usage request parameter

Indicates how the filter criteria is used. E.g. if this parameter is not provided, the Retrieve operation is for generic retrieve operation.

Table 6.3.6.11-1: Definition of Filter Usage

Value	Interpretation	Note
1	Discovery Criteria	
2	Event Notification Criteria	
NOTE:		

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 Functional Architecture [6]. Further details on the representation of primitives are specified in clauses 7.1.1.1 and 8.

Table 6.4.1-1: Data Types for Request primitive parameters

Primitive Parameter	Data Type	Multiplicity	Note
Operation	m2m:operation	1	See Clause 6.3.3.2.5
To	xs:anyURI	1	
From	m2m:ID	1	See Clause 6.3.2
Request Identifier	m2m:requestID	1	See Clause 6.3.2
Resource Type	m2m:resourceType	0..1	See Clause 6.3.3.2.1
Name	xs:string	0..1	
Content	m2m:primitiveContent	0..1	See Clause 6.3.4.4
Originating Timestamp	m2m:timestamp	0..1	
Request Expiration Timestamp	m2m:timestamp	0..1	"Result Expiration Timestamp" shall be later than "Request Message Expiration Timestamp"
Result Expiration Timestamp	m2m:timestamp	0..1	
Operation Execution Time	m2m:timestamp	0..1	
Response Type	m2m:responseTypeInfo	0..1	See Clause 6.3.4.29
Result Persistence	xs:duration	0..1	
Result Content	m2m:resultContent	0..1	See Clause 6.3.3.2.7
Event Category	m2m:eventCat	0..1	See Clause 6.3.2
Delivery Aggregation	xs:boolean	0..1	
Group Request Identifier	xs:string	0..1	
Filter Criteria	m2m:filterCriteria	0..1	See Clause 6.3.4.7
Discovery Result Type	m2m:discResType	0..1	See Clause 6.3.3.2.8

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 TS-0001 Functional Architecture [6]. Further details on the representation of primitives are specified in clauses 7.1.1.1 and 8.

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 6.3.3.2.9
Request Identifier	m2m:requestID	1	See Clause 6.3.2
Content	m2m:primitiveContent	0..1	See Clause 6.3.4.4
To	m2m:ID	0..1	See Clause 6.3.2
From	m2m:ID	0..1	
Originating Timestamp	m2m:timestamp	0..1	See Table 6.3.2-1
Result Expiration Timestamp	m2m:timestamp	0..1	See Table 6.3.2-1
Event Category	m2m:eventCat	0..1	See Clause 6.3.2

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 TS-0001 Functional Architecture [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. In particular a resource might contain mandatory read-only primitives, and these would 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 TS0004 Core Protocol (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-v1_0_0.xsd, contain definitions for the common and universal attributes, and establish an inheritance hierarchy shown in Figure 6.5.1-1.

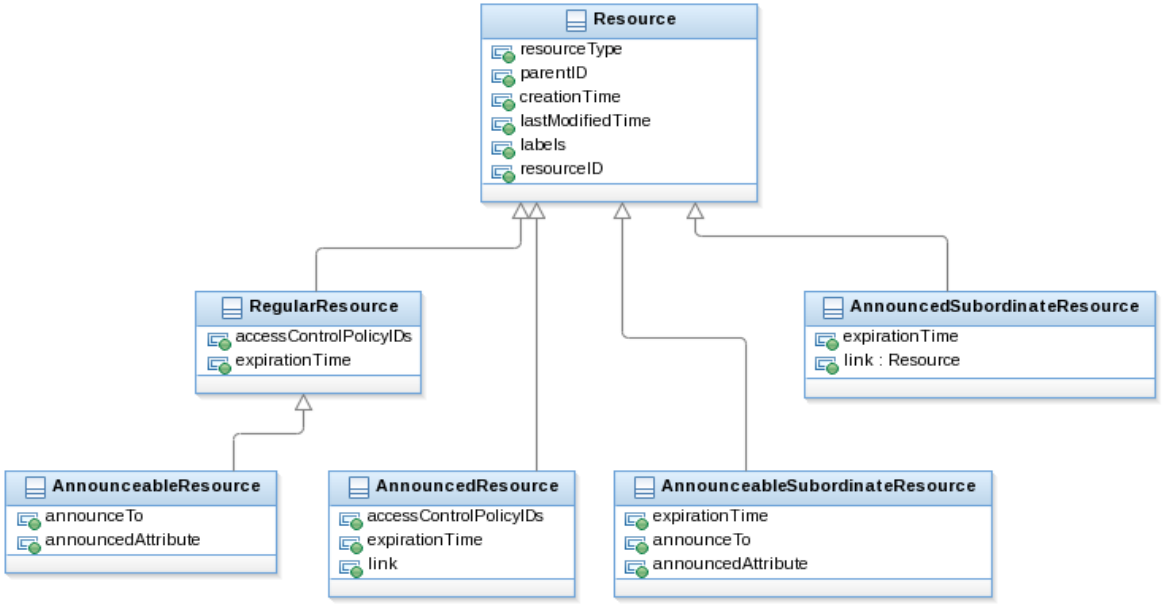

Figure 6.5.1-1: Resource Types

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 6.3.5-2

6.5.2.3 Usage

This type is used indirectly by all resource types. It is used directly just 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-annouceable M2M resources.

6.5.3.2 Reference

See Table 6.3.5-2.

6.5.3.3 Usage

This type is used by the following resource types:

<delivery>, <eventConfig>, <execInstance>, <m2mServiceSubscriptionProfile>, <mgmtCommand>, <pollingChannel>, <request>, <serviceSubscribedNode>, <statsCollect>, <statsConfig>, <subscription>, <serviceSubscribedAppRule>

6.5.4 announceableResource

6.5.4.1 Description

This type definition includes the universal and common attributes used by M2M 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 6.3.5-2.

6.5.4.3 Usage

This type is used by the following resource types:

AE>, <container>, <group>, <locationPolicy>, <node>, <remoteCSE>

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 6.3.5-2.

6.5.5.3 Usage

This type is used by the following resource types:

<AEAnnc>, <containerAnnc>, <groupAnnc>, <locationPolicyAnnc>, <nodeAnnc>, <remoteCSEAnnc>

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 6.3.5-2.

6.5.6.3 Usage

This type is used by the following resource types:

<AEAnnc>, <containerAnnc>, <groupAnnc>, <locationPolicyAnnc>, <nodeAnnc>, <remoteCSEAnnc>

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 6.3.5-2

6.5.7.3 Usage

This type is used by the following resource types:

<accessControlPolicyAnnc>, <contentInstanceAnnc>, <scheduleAnnc>.

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 categorised as one of 6 classes:

Table 6.6.2-1: Definition of Response Status Code class

Status Class	Codeclass	Interpretation
Informational	1xxx	The request is successfully received, but the request is still on process.
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 Serivce 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 innumeric code. The supplemental information may be returned when it is needed.

6.6.3.2 Informational response class

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

Table 6.6.3.2-1: Informational Responses class

Numeric Code	Description
1000	ACCEPTED

6.6.3.3 Successful response class

Table 6.6.3.2-1 specify 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	CHANGED

6.6.3.4 Redirection response class

Not defined any values in this response class.

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 specify the RSCs for Originator Error responses.

Table 6.6.3.5-1: RSCs for Originator Error response class

Numeric Code	Description
4000	BAD_REQUEST
4004	NOT_FOUND
4005	OPERATION_NOT_ALLOWED
4008	REQUEST_TIMEOUT
4101	SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE–
4102	CONTENTS_UNACCEPTABLE
4103	ACCESS_DENIED
4104	GROUP_REQUEST_IDENTIFIER_EXISTS
4105	CONFLICT

6.6.3.6 Receiver Error response class

Table 6.6.3.6-1 specify the RSCs for Receiver Error responses.

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	NO_PRIVILEGE
5106	ALREADY_EXISTS
5203	TARGET_NOT_SUBSCRIBABLE
5204	SUBSCRIPTION_VERIFICATION_INITIATION_FAILED
5205	SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE
5206	NON_BLOCKING_REQUEST_NOT_SUPPORTED

6.6.3.7 Network System Error response class

Table 6.6.3.7-1 specify the RSCs for when the External System reported some errors.

Table 6.6.3.7-1: RSCs for Network Service Error response class

Numeric Code	Description
6003	EXTENAL_OBJECT_NOT_REACHABLE
6005	EXTENAL_OBJECT_NOT_FOUND
6010	MAX_NUMBERF_OF_MEMBER_EXCEEDED
6011	MEMBER_TYPE_INCONSISTENT
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_CANCELATION_FAILURE
6028	ALREADY_COMPLETE
6029	COMMAND_NOT_CANCELLABLE

6.7 oneM2M specific MIME media types

The present sub-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 "application" mediate 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 oneM2M resource in content shall be indicated by "ty" parameter. XML serialization rule is applied. (See clause 7.4.2)
vnd.onem2m-res+json	m2m:resource	Same information of above. JSON serialization rule is applied. (See clause 7.4.2)
vnd.onem2m-ntfy+xml	m2m:notification or m2m:aggregatedNotification	For Notify operation for resource subscription. XML serialization rule is applied. (See clause 7.4.1)
vnd.onem2m-ntfy+json	m2m: notification or m2m:aggregatedNotification	Same information of above. JSON serialization rule is applied. (See clause 7.4.1)
vnd.onem2m-attrs+xml	m2m:attributeList	For exchanging alist of oneM2M resource attributes and its value when it is needed. XML serialization rules is applied. (See clause 7.4.2)
vnd.onem2m-attrs+json	m2m:attributeList	Same information of above. JSON serialization rule is applied. (See clause 7.4.2)
vnd.onem2m-preq+xml	m2m:requestPrimitive	For exchanging serialized oneM2M request primitive. XML serialization rule is applied. (See clause 6.4.1 and 7.1.1.1)
vnd.onem2m-preq+json	m2m:requestPrimitive	Same information of above. JSON serialization rule is applied. (See clause 6.4.1 and 7.1.1.1)
vnd.onem2m-prsp+xml	m2m:responsePrimitive	For exchanging Response parameters. XML serialization rules is applied. (See clause 6.4.2 and 7.1.1.2)
vnd.onem2m-prsp+json	m2m:responsePrimitive	Same information of above. JSON serialization rule is applied. (See clause 6.4.2 and 7.1.1.2)

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>	latest	<container>	See clause 7.3.27
<oldest>	oldest	<container>	See clause 7.3.28
<fanOutPoint>	fanOutPoint	<group>	See clause 7.3.14
<pollingChannelURI>	pollingChannelURI	<pollingChannel>	See clause 7.3.22

Each resource instance listed in “Parent Resource” column of Table 6.8-1has 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.

The parent resources contain named references, whose names match the virtual child’s resourceNames. Each reference is a URI to the corresponding virtual resource. In the <container> case, there are two such references, one called *latest* and one called *oldest*. The URI returned stays valid for the lifetime of the virtual resource.

A virtual resource can also be addressed using a hierarchical URI formed by taking the hierarchical URI of the parent resource and appending a / followed by the resourceName of the virtual resource.

7 oneM2M procedures

The following clauses describe prerequisites such as primitive format and procedure outline with three generic scenarios that are Originator, Receiver, and Resource Handling in accordance with CRUD+N operations. In addition, for specific resource type they provide common or resource specific attributes, data type definition for the attributes, and child resources as well as they explain resource specific procedures on CRUD operations to communicate with oneM2M compliant M2M Platform System by oneM2M protocols and APIs as follows:

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

7.1 Primitive format and generic procedure

7.1.1 Primitive format

7.1.1.1 Request primitive format

Table 7.1.1.1-1 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 7.1.1.1-1: Request Primitive Parameters

Primitive Parameter	CREATE	RETRIEVE	UPDATE	DELETE	NOTIFY
Operation	M	M	M	M	M
To	M	M	M	M	M
From	M	M	M	M	M
Request Identifier	M	M	M	M	M
Resource Type	M	NP	NP	NP	NP
Name	O	NP	NP	NP	NP
Content	M	O	M	NP	M
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

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

1) A complete or partial Resource. In this case the Content shall contain a single 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 7.3, however if a partial resource is being transferred, it is not required to be valid according to the XSD for that resource.

2) A Notification Data Object. This is named <m2m:notification> and is described in Clause 7.4.1

3) An Aggregated Notification. This is named <m2m:aggregatedNotification> and contains multiple <m2m:notification> objects. This is described in clause 7.4.1.

4) An AttributeList element, as described in clause 7.4.2. This is used in the partial retrieval case to pass a set of attribute Names (by leaving the attribute values blank). In the partial update case it is used to pass a set of attribute Name/Value pairs.

5) A ResponsePrimitive object as described in clause 7.4.1. This is used in Asynchronous non-blocking case.

7.1.1.2 Response primitive format

Table7.1.1.2-1 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 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.

Table7.1.1.2-1 : 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

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

1) A complete or partial Resource. In this case the Content shall contain a single 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 7.3, however if a partial resource is being transferred, it is not required to be valid according to the XSD for 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 an element called m2m:resource defined in clause 7.4.2. The URI is included as an attribute of m2m:resource.

4) A list of URIs. This can be used for transferring the childResource URIs only or in a Discovery response. These are included in an element called m2m:URIList defined in clause 7.4.2.

5) An Aggregated Response. This is sent as a result of a Group operation. This uses the element m2m:aggregatedResponse defined in clause 7.4.2.

6) Raw data. This could be a simple data value, or structured data (an XML complex type or JSON object)

7.1.2 Description of generic procedures

7.1.2.1 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 sections by referencing these actions and providing additional steps. The Originator shall execute the following steps in order:

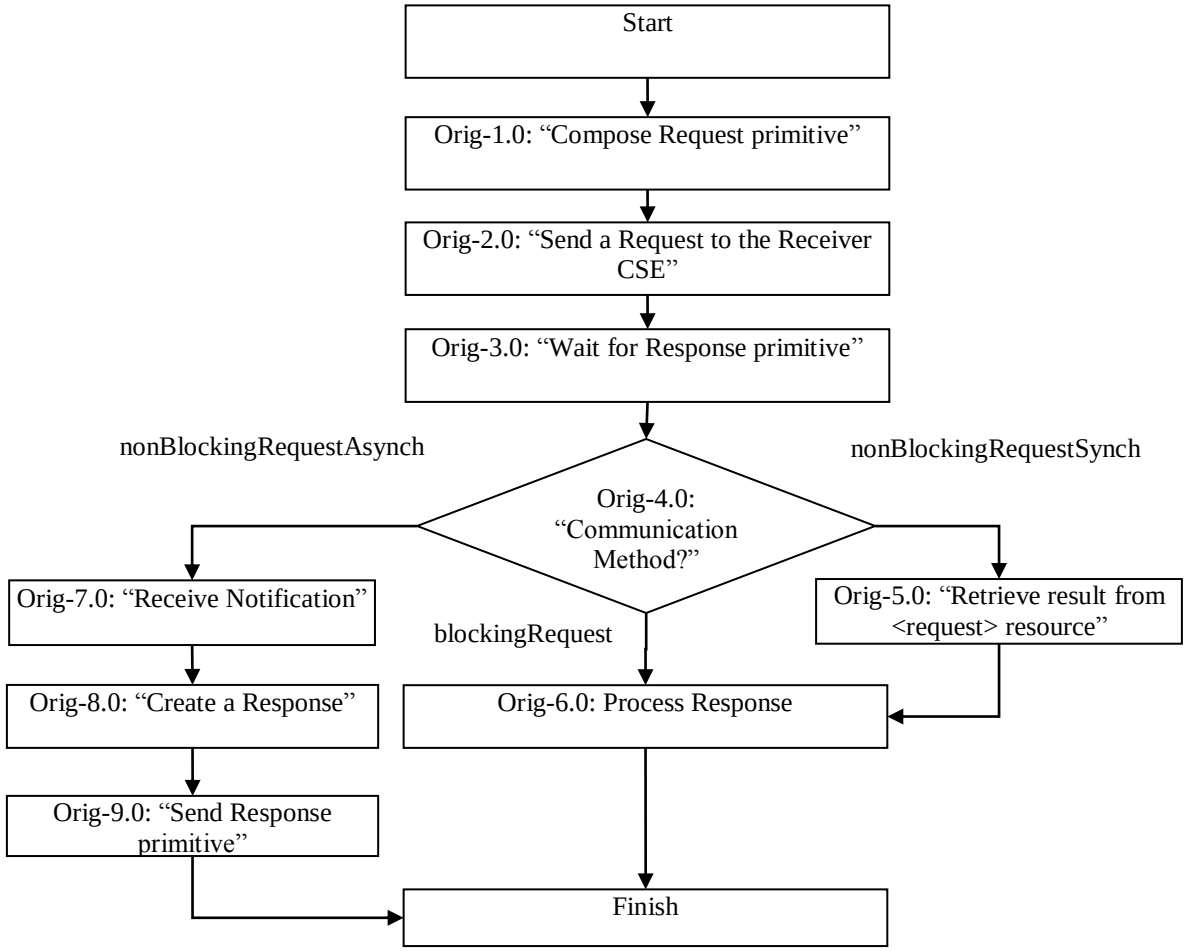

Figure 7.1.2.1-1: Generic procedure of Originator

Orig-1.0 "Compose Request primitive": Please refer to clause 7.2.1.1 for details.

Orig-2.0 "Send a Request to the Receiver CSE": Please refer to clause 7.2.1.2 for details.

Orig-3.0 "Wait for Response primitive": Please refer to clause 7.2.1.3 for details.

Orig-4.0 "Communication Method?": This step shall be operated after getting the Response primitive from step Oring-3.0 "Wait for Response primitive". In this step, the Originator checks whether the request was blockingRequest, nonBlockingRequestSynch or nonBlockingRequestAsynch by using ***Response Type*** parameter (see detail in clause 8.1.2 in the oneM2M TS-0001 Functional Architecture [6]).

If the request was blockingRequest it goes to step Orig-6.0 "Process Response". If the request was nonBlockingRequestSync, it goes to step Orig-5.0 "Retrieve result from the <request> resource". If the request was nonBlockingRequestAsynch, it goes to step Orig-7.0 "Receive Notification".

Orig-5.0 "Retrieve result from the <request> resource": See clause 7.2.1.4 for details.

Orig-6.0 "Process Response": the Originator processes the response.

Orig-7.0 "Receive Notification": the Originator processes the notification.

Orig-8.0 “Create a Response”: Please refer to clause 7.2.2.2 for details.

Orig-9.0 “Send Response primitive”: Please refer to clause 7.2.2.3 for details..

7.1.2.2 Generic request procedure for 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 7.2.3.12 for details) and then "Send Response primitive" (refer to clause 7.2.2.4for details). The corresponding Response code shall be included in the Response primitive.

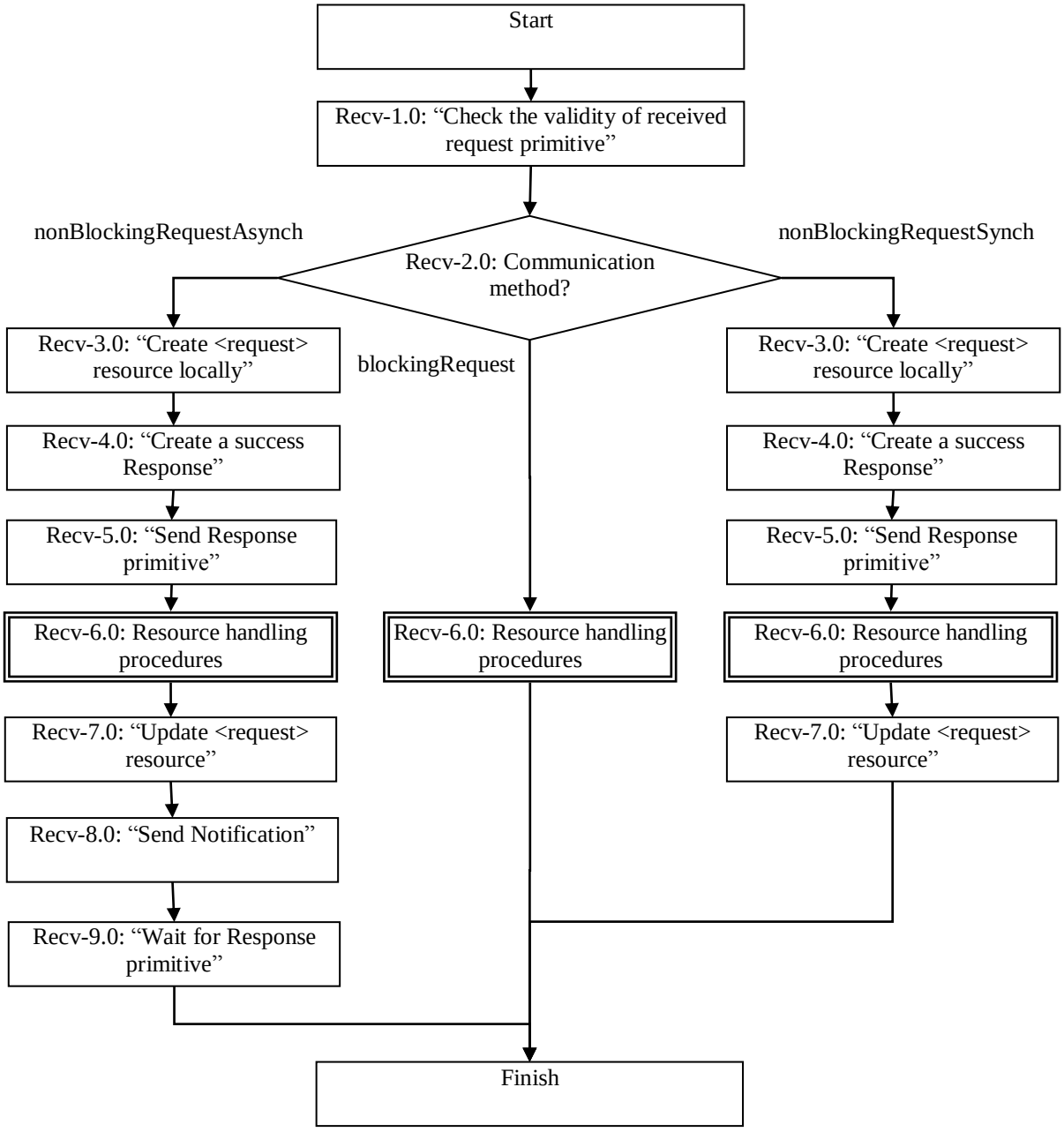

Figure 7.1.2.2-1: Generic procedure of Receiver

Recv-1.0 “Check the validity of received request primitive”: See clause 7.2.2.1 for details.

Recv-2.0 “Communication method?”: The Receiver CSE checks whether a received request is blockingRequest, nonBlockingRequestSynch or nonBlockingRequestAsynch by using ***Response Type*** parameter (see detail in clause

8.1.2 in TS-0001 Functional Architecture [6]). If the request is blockingRequest or ***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”.

Recv-3.0 “Create <request> resource locally”: Please refer to clause 7.2.2.2 for details.

Recv-4.0 “Create a successResponse”: Please refer to clause 7.2.2.2 for details.

Recv-5.0 “Send Response Primitive”: Please refer to clause 7.2.2.4 for details.

Recv-6.0 “Resource handling procedure”: Please refer to

Figure 6.3.3.2.30-2for details.

Recv-7.0 “Update <request> resource”: Please refer to clause 7.2.2.5 for details. This step is only valid when the request is non-blocking.

Recv-8.0 “Send Notification”: Please refer to clause 7.4.1.2.4 for details.

Recv-9.0 “Wait for a Response primitive”: Please refer to clause 7.2.1.3 for details.

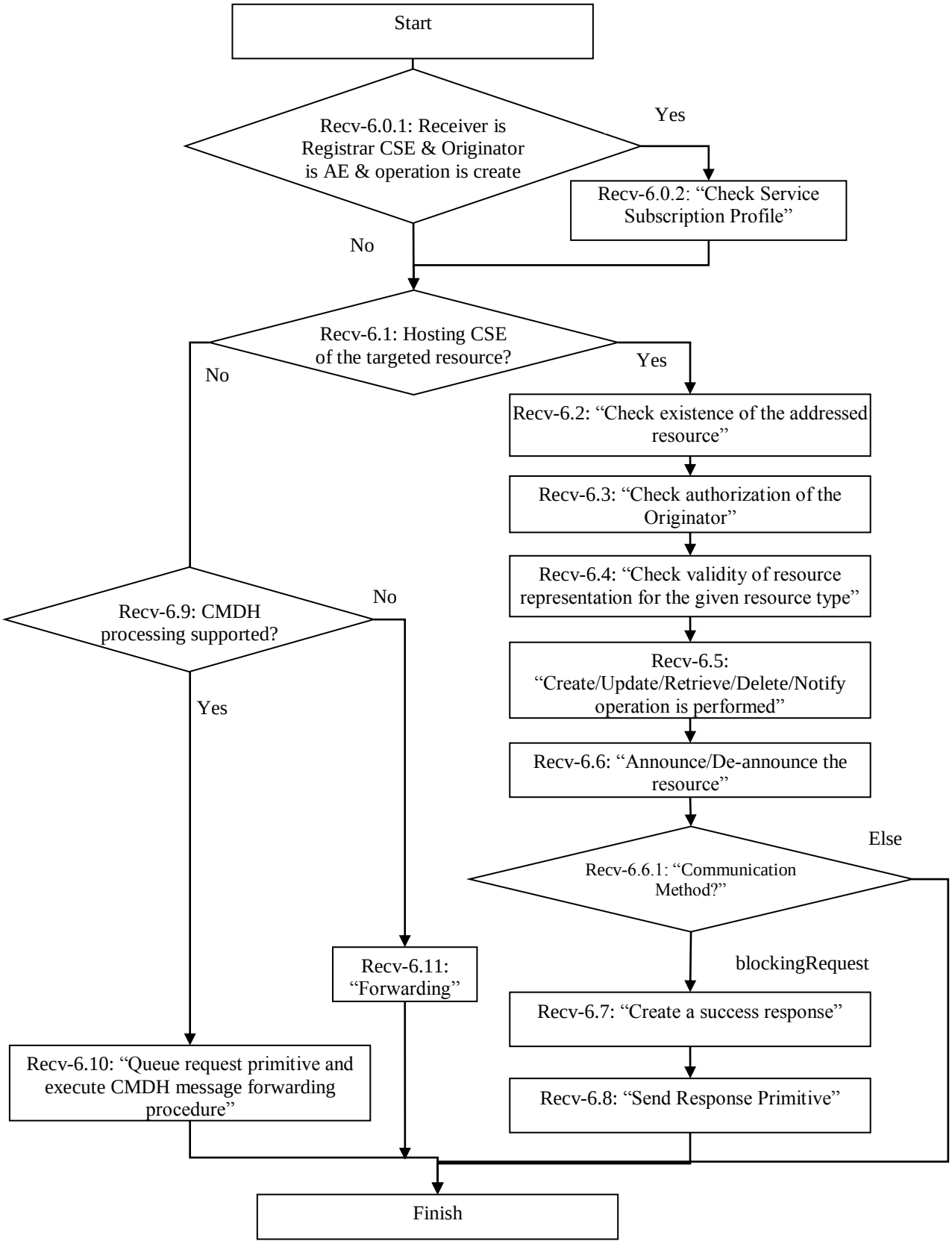

Figure 6.3.3.2.30-2: Resource handling procedure

The above figure describes the generic procedure to resource handling procedures.

Recv-6.0.1 “Receiver is Registrar CSE, Originator is AE and operation is create?”: The step checks if the receiver is Registrar CSE, the Originator is AE, and operation is create. If the receiver is Registrar CSE and Originator is an AE, goes to Recv-6.0.2 “Check Service Subscription Profile”. Otherwise, goes to Recv-6.1.

Recv-6.0.2 “Check Service Subscription Profile”: Please refer to clause 7.2.2.7 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 “Queue request primitive and execute CMDH message forwarding procedure”, No branch).

Recv-6.2 “Check existence of the addressed resource”: Please refer to clause 7.2.3.1 for details.

Recv-6.3 “Check authorization of the Originator”: Please refer to clause 7.2.3.14 for details.

Recv-6.4 “Check validity of resource representation”: Please refer to clause 7.2.3.2 and clause 7.2.3.3 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 7.2.3.4)”, “Retrieve the resource (clause 7.2.3.5)”, “Update the resource (clause 7.2.3.6)”, “Delete the resource (clause 7.2.3.7)” and “Notify re-targeting (clause 7.2.3.8)”. Notify re-targeting is performed for the Create, Update, Retrieve, Delete, or Notify operation respectively.

Recv-6.6 “Announce/De-announce the resource”: The step represents two common operations which are “Announce the resource” and “De-announce the resource”. Please refer to clause 7.2.3.9 and clause 7.2.3.10 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 TS-0001 Functional Architecture [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 7.1.1.2.2-1).

Recv-6.7 “Create a success response”: Please refer to clause 7.2.3.11 for details.

Recv-6.8 “Send Response Primitive”: Please refer to clause 7.2.2.4 for details. If the Receiver is Hosting CSE, after this step, the procedure is terminated.

Recv-6.9 “CMDH processing supported?”: This step checks whether the Receiver supports the CMDH processing.

Recv-6.10 “Queue request primitive and execute CMDH message forwarding procedure”: If CMDH message is supported, the Receiver CSE shall queue the received request primitive and execute the “CMDH message forwarding procedure”. Please refer to Annex H.2.4. for details.

Recv-6.11 “Forwarding”: If CMDH processing is not supported, carry out message forwarding as defined in clause 7.2.2.6.

7.2 Common operations

7.2.1 Originator actions

7.2.1.1 Compose request primitive

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

The Request shall include the ***From*** and ***To*** parameters to indicate the identifier of the originator of the request and the targeted receiver of the request.

The Request shall include the other attributes in case needed depend on the resource the request is addressing.

When including a resource representation in the request indication 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 indication.

7.2.1.2 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.

7.2.1.3 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.

7.2.1.4 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 7.2.1.1 and 7.2.1.2.

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

Table 7.2.1.4-1: 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.

7.2.2 Receiver CSE actions

7.2.2.1 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 [2322], and MQTT binding [24]). The received resource representation (e.g. in plain XML, binary XML or JSON) shall be validated against the provided schema definitions.

If the received request is communicated within an established Security Association (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 "ACCESS_DENIED" ***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 "ACCESS_DENIED" ***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 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 a "ACCESS_DENIED" ***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 an "ACCESS_DENIED" ***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 Annex 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.

7.2.2.2 Create <request> resource locally

Creation of a <request> resource can only be done on a Receiver CSE implicitly. When the Receiver CSE receives a request for targeting any other resource type or requesting a notification in non-blocking mode, i.e. the ***Response Type*** parameter of the request is set to either 'nonBlockingRequestSynch' or 'nonBlockingRequestAsynch', 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_REQUEST_NOT_SUPPORTED" error. For the 'nonBlockingRequestAsynch'request, a Receiver CSE that does not support the <request> resource type shall be able to 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 a value to the common attributes of <request> resource according to the following table:

Table 6.3.3.2.30-1: 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.
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 one ID of an <accessControlPolicy> that contains the following: In the '***privileges***' attribute • Allow RUD operations to the Hosting CSE • Allow RD operations to the Originator, i.e. the value of the parameter ***From*** in the associated non-blocking request In the '***selfPrivileges***' attribute 1. Allow U operations the parent <accessControlPolicy> resource to the Originator, i.e. the value of the parameter ***From*** in the associated non-blocking request
labels	Originator ID
stateTag	0
resorceName	Hosting CSE shall assign a value to this attribute.

2) Assign a value to the resource-specific attributes of <request> resource according to the following table:

Table 6.3.3.2.30-2: 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 [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

7.2.2.3 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 6.3.3.2.30-1: 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.

7.2.2.4 Send response primitive (acknowledgement)

A Response primitive shall be sent back to the originator.

7.2.2.5 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 <request> resource based on the following steps.

1) Update a value to the common attributes of <request> resource according to the following table:

Table 6.3.3.2.30-1: Common attributes settings for <request> resource

Attribute Name	Value
lastModifiedTime	Date/time of the last modification.
stateTag	This value is incremented on every modification.

2) Update a value to the resource-specific attributes of <request> resource according to the following table:

Table 6.3.3.2.30-2: 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	Hosting CSE shall contain the response message of the originally requested operation – if any – in line with the ***rc*** parameter in the associated non-blocking request.

7.2.2.6 Forwarding

If the ***To*** parameter in the request does not start with the CSEBase URI of the receiver, the receiver CSE shall forward the request or shall serve the request locally (see below).

If the ***To*** parameter in the request starts with the CSEBase URI of the receiver, then the receiver CSE shall handle the request locally.

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

1) "Send a Request to the receiver CSE".

2) "Wait for Response primitive".

When the Response is received the receiver CSE shall:

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

7.2.2.7 Check Service Subscription Profile

The receiver shall check if the originator's <*serviceSubscriptionProfile*> has S-RoleID in *serviceRoles* attribute that allows to create resource type specified in resource type parameter. The receiver shall find a proper <*serviceSubscribedNode*> including CSE-ID of the receiver and determine serviceRoles from the parent resource of the <*serviceSubscribedNode*> which corresponds to <*serviceSubscriptionProfile*> resource.

7.2.3 Hosting CSE actions

7.2.3.1 Check existence of the addressed resource

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.

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

The ***resourceName*** attribute has special handling. If the ***resourceName***-attribute is present in the CREATE request, the hosting CSE shall check if a resource with the same resourceName already exists in the addressed collection. 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 assign an resourceName to the resource being created.

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.

M attribute

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

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

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.

7.2.3.3 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.

M attribute

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.

If the attribute is not 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.

O attribute

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

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 unless the value provided for the attribute exactly matches the value in the current resource representation stored in the hosting CSE. In addition, the lastModifiedTime attribute shall always be accepted (but ignored) by the hosting CSE, no matter what value was provided in the request.

7.2.3.4 Create the resource

A new resource shall be created and correlated to the addressed and existing parent resource. As the result of the resource creation, the ***lastModifiedTime*** attribute of the parent resource shall be set to the same value as the ***creationTime*** attribute of the created resource. The following rules shall be applied.

The URI of the created resource shall be the URI of its parent resource with the resourceName appended. (e.g. http://CSEbase.operator.org/myAppID, for an application resource with resourceName "myAppID" created in the parent resource http://CSEbase.operator.org).

If a resource with the same resourceName already exists among the siblings of the addressed parent resource, the hosting CSE shall provide a new resourceName that is unique within the parent.

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

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

M attribute

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

O attribute

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

If 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 7.3.

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.

NP attribute

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.

7.2.3.5 Retrieve the resource

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 the 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.

7.2.3.6 Update the resource

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 7.2.3.9 and clause 7.2.3.10, respectively.

M attribute

If the provided attribute value is accepted, the server shall use the provided value in the resource representation of the updated resource.

O attribute

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

If the attribute is not provided or accepted, but the multiplicity of the attribute is "1", the hosting CSE shall assign a default value or assign a value based on local policy, or the value specified in this specification in clause 7.3.

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 hosting CSE shall delete such attribute if the deletion of the attribute is allowed by the local policy.

NP attribute

If the attribute is not present in the resource representation in the UPDATE request and the multiplicity of the attribute is "1" in the resource, then the hosting CSE shall not update the attribute value. There is 2 exceptions to this rule and they are the ***lastModifiedTime*** attribute and ***stateTag*** attribute. The hosting CSE shall set the lastModifiedTime to the current time whenever an update primitive is received. The hosting CSE shall change the stateTag each time an update primitive is received.

If the attribute is present in the resource representation in the UPDATE request the presented value shall be ignored, i.e. the hosting CSE shall never update its resource representation based on the presence of an NP attribute value in an update.

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 by "Delete the addressed resource".

7.2.3.7 Delete the resource

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 parent resource of the addressed resource shall be updated to remove the reference to the deleted resource. If the parent resource has a ***lastModificationTime*** attribute then this attribute shall be set to the time of the deletion.

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

7.2.3.8 Notify re-targeting

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 ***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 pointOfAccess value

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

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

7.2.3.9 Announce the resource or attribute

If CREATE request that contains an ***announceTo*** attribute is received,

- Compose the CREATE Request primitive as follows:
  - Link is set to the URI of the original resource.
  - If accessControlPolicyIDs of the original resource is not present, accessControlPolicyIDs is set to the same value with the parent resource or from 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.
- Send a CREATE Request to the CSE(s) represented by exact URI(s) or CSE-ID(s) in the announceTo of the request.
- Wait for Response primitive
- Add the URI of successfully announced resource to the ***announceTo*** attribute of the resource

- Include updated ***announceTo*** attribute in the ***Content*** parameter in the Response to the received CREATE Request.

If UPDATE request that adds the URI or CSE-ID into the ***announceTo*** attribute is received,

- Compose the CREATE Request primitive as follows:
  - Link is set to the URI of the original resource.
  - If accessControlPolicyIDs of the original resource is not present, accessControlPolicyIDs is set to the same value with the parent resource or from 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.
- Send a CREATE Request to the CSE(s) represented by exact URI(s) or CSE-ID(s) in the announceTo of the request, which is not included in the announceTo attribute of the original resource.
- Wait for Response primitive
- Add the URI of successfully announced resource to the ***announceTo*** attribute of the resource
- Include updated ***announceTo*** attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

If UPDATE request that adds the attribute name into the ***announcedAttribute*** attribute is received,

- Compose the UPDATE Request. 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.
- Send UPDATE Requests to all announced resources listed in the ***announceTo*** attribute.
- Wait for Response primitive.
- Add the attribute name of the successfully announced attribute to the ***announcedAttribute*** attribute.
- Include updated ***announcedAttribute*** attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

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

- Compose an UPDATE Request primitive by including the updated attribute(s) with its associated updated value.
- Send the UPDATE Request to all CSE(s) represented by the URI(s) in the ***announceTo*** attribute of the original resource.

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

- Compose an UPDATE Request primitive by including the updated attribute(s) with its value set to NULL.
- Send the UPDATE Request to all CSE(s) represented by the URI(s) in the ***announceTo*** attribute of the original resource.

7.2.3.10 De-announce the resource or attribute

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

- Compose the DELETE Request primitive.

- 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.
- Wait for Response primitive.
- Remove the URI of successfully de-announced resource from the ***announceTo*** attribute of the resource.
- Include updated ***announceTo*** attribute in the ***Content*** parameter in the Response to the UPDATE Request of the original resource.

If DELETE Request is received:

- Compose the DELETE Request primitive.
- Send DELETE Requests to all announced resources addressed by the URI(s) in the ***announceTo*** attribute of the resource.
- Wait for Response primitive.

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

- 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.
- Send UPDATE Requests to all announced resources listed in the ***announceTo*** attribute of the original resource.
- Wait for Response primitive.
- Delete the attribute name of the successfully de-announced attribute from the ***announcedAttribute*** attribute.
- Include updated ***announcedAttribute*** attribute in the ***Content*** parameter in the Response to the received UPDATE Request.

7.2.3.11 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. If the Hosting CSE assigned attribute(s) not provided or modified any of the provided attributes as provided in the Request, the *Content* parameter shall include the assigned and/or modified attributes;
- "OK" in case of Retrieve operation;
- "OK" in case of Update operation;
- "OK" 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 Hosting CSE shall include *Content* parameter with:

- the address and/or attributes(assigned and modified by the Hosting CSE) of the created reseource depending on Result Content parameter (i.e., attributes, hierarchical-address, hierarchical-address+attributes) in the request primitive. This shall apply for Create operation;
- the retrieved attributes and/or child resource references depending on Result Content parameter (i.e., attributes, attributes+child-resources, attributes+child-resource-references, child-resource-references, original-resource) in the request primitive. This shall apply for Retrieve operation; and
- the modified/created/deleted attributes. This shall apply for Update operation.

More details can be found in clause 7.1.1.2 (Response primitive format).

NOTE: If Result Content parameter is not given in the request primitive, the default value is attributes. How to deal with each Result Content value is described in clause 8.1.2 [6]).

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

7.2.3.12 Create an error response

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

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

7.2.3.13 Resource discovery procedure

A resource discovery is used to discover resources in a CSE. A Resource discovery request is done by sending Retrieve request with *filterUsage*, one of the ***filterCriteria*** parameters, configured as "discovery" and the request may include other ***filterCriteria*** 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 7.1.2.1 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 ***filterUsage*** parameter in ***filterCriteria***.

The Retrieve Request may include other parameters of ***filterCriteria***.

Receiver:

The Receiver shall follow the steps from Recv-1.0 to Recv-7.0 specified in clause 7.1.2.2 Generic Resource Request Procedure for Receiver.

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

The Receiver shall find resources, which match all the configured *filterCriteria* and which the Originator has "Discover" access right, under the addressed resource".

In Recv-6.7, the Receiver shall include addresses for all the found resources.

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

7.2.3.14 Check authorization of the originator

Depending on the target resource type, the Hosting CSE shall use *accessControlPolicyIDs* of the different resources.

- For <schedule> resource, the Hosting CSE shall evaluate the *accessControlPolicyIDs* of the parent resource.
- For <latest>, <oldest> and <contentInstance> resource, the Hosting CSE shall evaluate the *accessControlPolicyIDs* of the parent <container> resource.
- For <m2mServiceSubscriptionProfile> and <serviceSubscriedNode> resource, if it has no *accessControlPolicyIDs* value, the Hosting CSE shall evaluate the *accessControlPolicyIDs* of the parent resource.
- For other resources, the Hosting CSE shall evaluate the *accessControlPolicyIDs* of the resource.

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 it is failed. For more details, see the clause 7.1.5 in TS-0003 Security Solutions [7].

- *accessControlOriginators* of the rule includes the Originator information.
- *accessControlContexts* of the rule includes the request context, if the rule includes the *accessControlContexts*
- *accessControlOperations* of the rule matches the operation type of the request.

If the evaluation failed, then authorization failure information shall be returned to the Originator.

7.2.4 Management common operations

7.2.4.1 Identify the managed entity and the management protocol

The Hosting CSE shall identify the managed entity to be managed via the <node> resource which is the parent resource in case of an addressed <mgmtObj> resource. In case of a <mgmtCmd> resource the entity to be managed is indicated in the ***execTarget*** attribute which addresses either a <node> resource or a group of resources of type <node>. Hence, in all cases the managed entity is ultimately identified through the <node> resource, from which the identifier of the device can be retrieved.

Then the Hosting CSE shall determine the management protocol to be used for communicating with the managed entity based on the objectID 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.

7.2.4.2 Locate the external management objects to be managed on the managed entity

The Hosting CSE shall locate the external management object information 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 external management object information on the managed entity through the ***objectPaths*** attribute of the <mgmtObj> resource of the addressed [objectAttribute], combined with their relative position in the external management object tree. If the external management object information 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 external management 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.

7.2.4.3 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 management protocol in case such triggering mechanism is supported by the external management technology. If the triggering mechanism is not supported by the external management technology, 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 external management objects, the Hosting CSE shall establish a session with the managed entity with the necessary access control privileges to perform the management request on the external management 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.

7.2.4.4 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 external management object information on the managed entity as determined in clause 7.2.4 or in the primitive specific clauses. The management request being used is specific to the external management technology according to a pre-defined mapping relationship with the Request primitive. The internal data structure of the external management object addressed by the management request shall be determined based on the mapping relationship of the <mgmtObj>, or <mgmtCmd> resources and the external management objects or based on the generic mapping rule as specified in TS-0001 Functional Architecture [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 external management technology.

7.3 Resource type-specific procedures and definitions

The reference point applicability of each resource-specific procedure for the following sub-clauses is described in the corresponding procedure specification in clause 10.2(Resource Type-Specific Procedures) [6]. E.g., Applicable reference points of <container> resource creation procedure (clause 7.3.6.2.1) in present specification is described as Mca, Mcc and Mcc' in clause 10.2.4.1(create <container> procedure) [6].

7.3.1 Resource type specification conventions

This clause describes how to understand the following clauses for resource type-specific procedures and definitions.

7.3.1.1 Resource type definition conventions

The following table includes the information of XSD data type definition files for the corresponding resource type.

Table 7.3.1.1-1: Data type definition of <resourceType>

Data Type ID	File Name	Note
Actual Data Type ID	*XSD file name*	

The following table includes the information of universal/common attributes of the resource type. Request optionality information means inclusion of the attribute name and its value in the request primitive is Mandatory(M)/Optional(O)/NP(Not Present). This is applicable for Create and Update operation only. For Retrieve operation, attribute names are optionally included in the request, but not with any values. For Delete operation, any attribute names or their values cannot be included in the request.

Universal/common attributes do not have any default value, however, have value restrictions and notes (see Table 6.3.5-1).

Table 7.3.1.1-2: Universal/Common Attributes of <resourceType> resource

Attribute Name	Request Optionality	
	Create	Update
Universal/common attribute name	*M/O/NP*	*O/NP*

The following table includes the information of resource specific attributes of the resource type. Convention for request optionality is the same as the universal/common attribute table above.

Table 7.3.1.1-3: 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*		

The following table includes the information of child resources of the resource type.

Table 7.3.1.1-4: 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 this TS*

7.3.1.2 Resource type-specific procedure conventions

This clause describes resource type specific procedures referring generic procedures defined in clause 7.1.2. 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 7.1.2.

If a deviation/addition procedure includes sub-procedures in one more more level(s), proper numbering is used to show the levels (e.g., "1)", "a)"). If sub-procedures do not care the order, bullets are used instead of numbers

7.3.2 Resource type <accessControlPolicy>

7.3.2.1 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 accessContolOperations) 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 TS-0001 Functional Architecture [6].

Table 7.3.2.1-1: Data type defintion of <accessControlPolicy> resource

Data Type ID	File Name	Note
accessControlPolicy	CDT-accessControlPolicy-V1_0_0.xsd	

Table 7.3.2.1-2: Universal/Common Attributes of <accessControlPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	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 7.3.2.1-3: 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

The following table includes the information of child resources of the resource type.

Table 7.3.2.1-4: Child Resources of <accessControlPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.2.2 accessControlPolicy resource specific procedure on CRUD operations

This sub-clause describes accessControlPolicy resource specific behaviour for CRUD operations.

7.3.2.2.1 Create

Originator:

No changes from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.2.2.2 Retrieve

Originator:

No changes from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.2.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.2.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.3 Resource Type <CSEBase>

7.3.3.1 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 TS-0001 Functional Architecture [6]).

Table 7.3.3.1-1: Data type definition of <CSEBase> resource

Data Type ID	File Name	Note
CSEBase	*CDT-CSEBase-v1_0_0.xsd*	

Table 7.3.3.1-2: Universal/Common Attributes of <CSEBase> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
Labels	NP	NP

Table 7.3.3.1-3: Resource Specific Attributes of <CSEBase> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
cseType	NP	NP	m2m:cseTypeID	No default
CSE-ID	NP	NP	m2m:ID	No default
supportedResource Type	NP	NP	list of m2m:resourceType	No default
pointOfAccess	NP	NP	m2m:pOAList	No default
nodeLink	NP	NP	xs:anyURI	No default

Table 7.3.3.1-4: Child resources of <CSEBase> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<remoteCSE>	[variable]	0..n	Clause 7.3.4
<node>	[variable]	0..n	Clause 7.3.18
<AE>	[variable]	0..n	Clause 7.3.5
<container>	[variable]	0..n	Clause 7.3.6
<group>	[variable]	0..n	Clause 7.3.13
<accessControlPolicy>	[variable]	0..n	Clause 7.3.2
<subscription>	[variable]	0..n	Clause 7.3.8
<mgmtCmd>	[variable]	0..n	Clause 7.3.16
<locationPolicy>	[variable]	0..n	Clause 7.3.10
<statsConfig>	[variable]	0..n	Clause 7.3.23
<statsCollect>	[variable]	0..n	Clause 7.3.25
<request>	[variable]	0..n	Clause 7.3.12
<delivery>	[variable]	0..n	Clause 7.3.11
<schedule>	[variable]	0..1	Clause 7.3.9
<m2mServiceSubscriptionPolicy	[variable]	0..n	Clause 7.3.19
<serviceSubscribedAppRule>	[variable]	0..n	Clause 7.3.29

7.3.3.2 <CSEBase> resource specific procedure on CRUD operations

7.3.3.2.1 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".

7.3.3.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.3.2.3 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 .

e) "Send the Response primitive".

7.3.3.2.4 Delete

Originator:

The <CSEBase> resource shall not be DELETEed 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.

f) "Send the Response primitive".

7.3.4 Resource Type <remoteCSE>

7.3.4.1 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>.

Conversely each registered CSE shall also be represented as a sub-set of <remoteCSE> resource in the registering CSE's <CSEBase>.

The detailed description can be found in clause 9.6.4 in Architecture TS.

Table 7.3.4.1-1: Data type definition of <remoteCSE> resource

Data Type ID	File Name	Note
remoteCSE	CDT-remoteCSE-v1_0_0.xsd	

Table 7.3.4.1-2: Universal/Common Attributes of <remoteCSE> resource

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	O	O
lastModifiedTime	NP	NP
labels	O	NP
announceTo	O	O
announcedAttribute	O	O

Table 7.3.4.1-3: 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	NP	NP	xs:anyURI	No default

Table 7.3.4.1-4: Child resources of <remoteCSE> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<AE>	[variable]	0..n	Clause 7.3.5
<container>	[variable]	0..n	Clause 7.3.6
<group>	[variable]	0..n	Clause 7.3.13
<accessControlPolicy>	[variable]	0..n	Clause 7.3.2
<subscription>	[variable]	0..n	Clause 7.3.8
<pollingChannel>	[variable]	0..n	Clause 7.3.21
<schedule>	[variable]	0..n	Clause 7.3.9

7.3.4.2 <remoteCSE> resource specific procedure on CRUD operations

The entire CSE registration procedure including <CSE> resource creation procedure below is defined in 10.1.1.2.1 [6] (“CSE registration procedure”).

7.3.4.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1 with the following exception:

- An AE shall not originate a Create <remoteCSE> resource request.

Receiver:

1) Primitive specific operation on Recv-1.0 "Check the syntax of received message":If the request is received over the Mca reference point, the Receiver CSE shall execute the following steps in order.

    a) "Create an unsuccessful Response primitive" with the response status code ‘OPERATION_NOT_ALLOWED’.

    b) "Send the Response primitive"

NOTE: Determination of the reference point is to the discretion of the Receiver CSE implementation.

7.3.4.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1.

7.3.4.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1.

7.3.4.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1.

7.3.5 Resource Type <AE>

7.3.5.1 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 TS-0001 Functional Architecture [6].

Table 7.3.5.1-1: Data type definition of <AE> resource

Data Type ID	File Name	Note
AE	CDT-AE-v1_0_0.xsd	XSD schema for AE resource

Table 7.3.5.1-2: Universal/Common Attributes of <AE> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	NP
announceTo	O	O
announcedAttribute	O	O

Table 7.3.5.1-3: 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	NP	NP	xs:anyURI	No default

Table 7.3.5.1-4: Child resources of <AE> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8
<container>	[variable]	0..n	Clause 7.3.6
<group>	[variable]	0..n	Clause 7.3.13
<accessControlPolicy>	[variable]	0..n	Clause 7.3.2
<pollingChannel>	[variable]	0..n	Clause 7.3.21

7.3.5.2 <AE> resource specific procedure on CRUD+N operations

This sub-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 the oneM2M TS-0001 [6] (“Application Entity registration procedure”).

7.3.5.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1 with the with the following exception:

- A CSE shall not originate a Create <AE> resource request.

Receiver:

Primitive specific operation on Recv-1.0 "Check the syntax of received message":

1. If the request is received over Mcc or Mcc’ reference point, the Receiver CSE shall execute the following steps in order.
   a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** ‘OPERATION_NOT_ALLOWED’.
   b) "Send the Response primitive".

NOTE:Determination of the reference point is to the discretion of the Receiver CSE implementation.

2. Otherwise,
   a) No change from the generic procedures in clause 7.1.2.2.

7.3.5.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.5.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.5.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.5.2.5 Notify

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.6 Resource Type <container>

7.3.6.1 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 TS-0001 Functional Architecture [6].

Table 7.3.6.1-1: Data type definition of <container> resource

Data Type ID	File Name	Note
container	CDT-container-V1_0_0.xsd	

Table 7.3.6.1-2: Universal/Common Attributes of <container> resource

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	O	O
lastModifiedTime	NP	NP
stateTag	NP	NP
labels	O	NP
announceTo	O	O
announcedAttribute	O	O

Table 7.3.6.1-3: Resource Specific Attributes of <contianer> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	NP	NP	m2m:ID	No default
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	O	O	xs:anyURI	No default
ontologyRef	O	O	xs:anyURI	No default

Table 7.3.6.1-4: Child resources of <container> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<contentInstance>	[variable]	0..n	Clause 7.3.7
<subscription>	[variable]	0..n	Clause 7.3.8
<container>	[variable]	0..n	Clause 7.3.6
<latest>	latest	1	Clause 7.3.27
<oldest>	oldest	1	Clause 7.3.28

7.3.6.2 <container> resource specific procedure on CRUD operations

This clause describes container resource specific behaviour for CRUD operations.

7.3.6.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.6.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.6.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

.

7.3.6.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.7 Resource Type <contentInstance>

7.3.7.1 Introduction

The <contentInstance> resource represents a data instance in the container.

The detailed description can be found in clause 9.6.7 in TS-0001 Functional Architecture [6].

Table 7.3.7.1-1: Data type definition of <contentInstance> resource

Data Type ID	File Name	Note
contentInstance	CDT-contentInstance-v1_0_0.xsd	

Table 7.3.7.1-2: Universal/Common Attributes of <contentInstance> 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
stateTag	NP	NP
Labels	O	NP
announceTo	O	NP
announcedAttribute	NP	NP

Table 7.3.7.1-3: Resource Specific Attributes of <contentInstance> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	NP	NP	m2m:ID	
contentInfo	M	NP	m2m:contentInfo	The default value of this attribute is 'text/plain:0'.
contentSize	O	NP	xs:nonNegativeInteger	No default
ontologyRef	O	NP	xs:anyURI	No default
content	M	NP	xs:anySimpleType	No default (Transfer encoding may be applied, and indicated applied encoding as part of the *contentInfo* attribute)

The ***contentInfo*** attribute shall provide meta information about the stored data in content. m2m:encodingType (0;plain, 1:base64 encoded string, 2:base64 encoded binary), and can be omitted when the Media Type of data is 'text/plain' and any transport encoding is not applied.

7.3.7.2 <contentInstance> resource specific procedure on CRUD operations

7.3.7.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2. The Originator may omit the name of the <contentInstance> resource unless the Originator need to refer specific content later.

7.3.7.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2. The Originator may omit the name of the targeted <contentInstance> resource when the latest version of stored content is requested.

7.3.7.2.3 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.

a) "Create an unsuccessful Response primitive" with the ***Response Status Code*** indicating "OPERATION_NOT_ALLOWED" error.

g) "Send the Response primitive".

7.3.7.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.8 Resource Type <subscription>

7.3.8.1 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 TS-0001 Functional Architecture [6].

Table 7.3.8.1-1: Data type definition of <subscription> resource

Data Type ID	File Name	Note
subscription	CDT-subscription-v1_0_0.xsd	

Table 7.3.8.1-2: Universal/Common Attributes of <subscription> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.8.1-3: Resource Specific Attributes of <subscription> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
eventNotificationCriteria	O	O	m2m:eventNotificationCriteria	No default
expirationCounter	O	O	xs:positiveInteger	No default
notificationURI	M	O	list of xs:anyURI	No default
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	No default
notificationEventCat	O	O	m2m:eventCat	No default
creator	O	O	m2m:ID	No default
subscriberURI	O	NP	xs:anyURI	No default

Table 7.3.8.1-4: Reference of child resources

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<schedule>	notificationSchedule	0..1	Clause 7.3.9

7.3.8.2 <subscription> resource specific procedure on CRUD operations

7.3.8.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

The following are additional Hosting CSE procedures to the generic resource handling procedures (Figure 7.1.2.2-1 in clause 7.1.2.2). The additional procedures shall be inserted from Recv-6.2 to Recv-6.8 as below.

The resource handling procedure for the Hosting CSE which receives <subscription> CREATE request shall perform the following procedures in order:

1. Recv-6.2
2. Recv-6.3
3. Check if the subscribed-to resource, addressed in ***To*** parameter in the Request, is subscribable. Subscribable resource types are defined in TS-0001 Functional Architecture [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.

4. 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 Notify response primitive with ***Response Status Code*** indicating "NO_PRIVILEGE" error.

5. If the *notificationURI* is not the Originator, the Hosting CSE should send a Notify request primitive to the *notificationURI* with ***verificationRequest*** parameter set as TRUE (See clause 7.4.1.2.2).

    a. If the Hosting CSE cannot send the Notify request primitive, the Hosting CSE shall return the Notify response primitive with a ***Response Status Code*** indicating "SUBSCRIPTION_VERIFICATION_INITIATION_FAILED" error.

    b. If the Hosting CSE sent the primitive, the Hosting CSE shall check if the Notify response primitive contains a ***Response Status Code*** indicating "SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE" or "SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE error. If so, the Hosting CSE shall return the Create response primitive with a ***Response Status Code*** indicating the same error from the Notify response primitive to the Originator.

6. Recv-6.4
7. Recv-6.5

   If the *notificationURI* is not the Originator, the Hosting CSE shall store Originator ID to ***creator*** attribute.

8. Recv-6.6
9. Recv-6.7
10. Recv-6.8

7.3.8.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.8.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.8.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.9 Resource Type <schedule>

7.3.9.1 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 Functional Architecture [6].

Table 7.3.9.1-1: Data type definition of <schedule> resource

Data Type ID	File Name	Note
schedule	CDT-schedule-V1_0_0.xsd	

Table 7.3.9.1-2: Universal/Common Attributes of <schedule> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	O	O
creationTime	O	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	NP
announceTo	O	O
announcedAttribute	O	O

Table 7.3.9.1-3: 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.

The ***scheduleElement*** attribute represents the list of scheduled tasks with the time of the execution.

The each entry of the ***scheduleElelement*** attribute shall consist of a line with 6 field values (See Table 7.3.8.1-4).

The time to be matched with the schedule pattern shall be interpreted in UTC timezone.

Table 7.3.9.1-4: 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

Each field value can be either an asterisk ('*': matching all valid values), an element, or an elements separated by commas(',').

An element shall be either a number or two numbers separated by a hyphen ('-': matching between two values).

The task which shall be executed is depending on the parent resource of the <schedule> resource (see Table 7.3.8.1-5).

Table 7.3.8.1-5: The task to be executed

Parent resource	Task to be executed	Note
<remoteCSE>	Establish connection to the remoteCSE	Timing of disconnection is up to implementation in present release.
<subscription>	Flash spooled notifications	

EXAMPLE 1:

EXAMPLE: * 0-5 2,6,10 * * *

In case of parent resource was <remoteCSE>, the CSE will be establish connection on 2:00-2:05, 6:00-6:05, and 10:00-10:05 every day.

End of EXAMPLE 1:

EXAMPLE 2:

EXAMPLE: * * 8-20 * * *

In case of the parent resource was <subscription>, the notification for the subscribed event will be suspended between from 20:00 to 8:00 on weekend.

End of EXAMPLE 2:

Table 7.3.9.1-5: Child resources of <schedule > resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
Subscription	[variable]	0..n	Clause 7.3.8

7.3.9.2 <schedule> resource specific procedure on CRUD operations

This sub-clause describes <schedule> resource specific behaviour for CRUD operations.

7.3.9.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

If <schedule> is created then scheduleElement (L) shall be created.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.9.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.9.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.9.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

If <schedule> is deleted then scheduleElement (L) shall be deleted.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.10 Resource Type <locationPolicy>

7.3.10.1 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 TS-0001 Functional Architecture [6] .

Table 7.3.10.1-1: Data type definition of <locationPolicy> resource

Data Type ID	File Name	Note
locationPolicy	CDT-locationPolicy-v1_0_0.xsd	

Table 7.3.10.1-2: Universal/Common Attributes of <locationPolicy> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	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

Table 7.3.10.1-3: 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
locationUpdatePeriod	O	O	xs:duration	No default
locationTargetID	O	NP	m2m:nodeID	No default
locationServer	O	NP	xs:anyURI	No default
locationContainerID	NP	NP	xs:anyURI	No default
locationContainerName	O	O	xs:string	No default
locationStatus	NP	NP	xs:string	No default

Table 7.3.10.1-4: Child resources of <locationPolicy> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.10.2 <locationPolicy> resource specific procedure on CRUD Operations

This clause describes <locationPolicy> resource specific primitive behaviour for CRUD operations.

7.3.10.2.1Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

The following <locationPolicy> resource type specific procedures shall be performed after Recv-6.5 and before Recv-6.6 generic procedures.

1) After the successful creation of <locationPolicy> resource, the Hosting CSE shall create <container> resource where the actual location information will be stored and the resource shall contain cross-references for the both resources, ***locationContainerID*** attribute for the <locationPolicy> resource and ***locationID*** attribute for the <container> resource. The name of the created <container> resource shall be determined by the ***locationContainerID*** 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), then continue with the step 3.

   h) 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.

   i) 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.

3) The Hosting CSE shall retrieve the ***locationTargetID*** and ***locationServer*** attributes from the stored <location Policy> resource.

   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 [28], 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 using, for example, OMA Mobile Location Protocol [i.4] and OMA RESTful NetAPI for Terminal Location [28]. 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: The detail information on how the Location Server request message is converted into OMA RESTful NetAPI for Terminal Location message is described in Annex G.

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 3: 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.

j) 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 is failed, the Hosting CSE shall store a statusCode based on the error code in the ***locationStatus*** attribute in the created <locationPolicy> resource.

k) 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.

7.3.10.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.10.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.10.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

The procedure of the Receiver written in the clause 7.1.2.2 (from *Rcv-D-1.0* to *Rcv-D-10.0*) 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 has been set with appropriate value, 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.

7.3.11 Resource Type <delivery>

7.3.11.1 Introduction

In order to be able to initiate and manage the execution of data delivery in a resource-based manner, 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 ON. The detailed description can be found in clause 9.6.11 in TS-0001 Functional Architecture [6].

Table 7.3.11.1-1: Data type definition of <delivery> resource

Data Type ID	File Name	Note
delivery	CDT-delivery-v1_0_0.xsd	

Table 7.3.11.1-2: Universal/Common Attributes of <delivery> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	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
stateTag	NP	NP

Table 7.3.11.1-3: 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 7.3.11.1-4: Child resources of <delivery> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	variable	0..n	Clause 7.3.8

7.3.11.2 <delivery> resource specific procedure on CRUD operations

This clause describes <delivery> resource specific behaviour for CRUD operations.

7.3.11.2.1 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 7.1.2.1.

***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 7.1.2.2.

NOTE: Determination of the reference point is to the discretion of the Receiver CSE implementation.

7.3.11.2.2 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 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.11.2.3 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 7.1.2.1.

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 7.1.2.2.

NOTE: Determination of the reference point is to the discretion of the Receiver CSE implementation.

7.3.11.2.4 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 7.1.2.1.

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 7.1.2.2

NOTE: Determination of the reference point is to the discretion of the Receiver CSE implementation.

7.3.12 Resource Type <request>

7.3.12.1 Introduction

The <request> resource is used to represent information on locally issued requests (i.e. issued by an AE or CSE internal). This allows for robust synchronous and asynchronous request processing coping with various constraints on maximum blocking time. When an AE or CSE issues a request for targeting any other resource type or requesting a notification in non-blocking mode , i.e. the ***Response Type*** parameter of the request is set to either '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. The detailed description can be found in clause 9.6.12 in oneM2M TS-0001 Architecture TS[6].

Table 7.3.12.1-1: Data type definition of <request> resource

Data Type ID	File Name	Note
request	*CDT-request-v1_0_0.xsd*	

Table 7.3.12.1-2: Universal/Common Attributes of <request> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
expirationTime	NP	NP
parentID	NP	NP
creationTime	NP	NP
lastModifiedTime	NP	NP
accessControlPolicy IDs	NP	NP
labels	NP	NP
stateTag	NP	NP

Table 7.3.12.1-3: 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
content	NP	NP	m2m:primitiveContent	No default
requestStatus	NP	NP	m2m:requestStatus	No default
operationResult	NP	NP	m2m:operationResult	No default

Table 7.3.12.1-4 : Reference of child resources

Child Resource Type Name	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.12.2 <request> resource specific procedure on CRUD operations

This clause describes request resource specific procedure on Resource Hosting CSE for CRUD operations.

7.3.12.2.1Create

Originator:

The <request> resource shall not be created via API. See clause 7.2.2.2 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".

7.3.12.2.2Retrieve

Originator: the procedure of the Originator is the same as the clause 7.1.2.1.

Receiver: the procedure of the Receiver is the same as the clause 7.1.2.2.

7.3.12.2.3Update

Originator:

The <request> resource shall not be updated via API. See clause 7.2.2.5 Update <request> resource.

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".

7.3.12.2.4Delete

Originator: the procedure of the Originator is the same as the clause 7.1.2.1

Receiver: the procedure of the Receiver is the same as the clause 7.1.2.2.

7.3.13 Resource Type <group>

7.3.13.1 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 ***membersID*** 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 TS-0001 Functional Architecture [6].

Table 7.3.13.1-1: Data type definition of <group> resource

Data Type ID	File Name	Note
group	CDT-group-v1_0_0.xsd	

Table 7.3.13.1-2: Universal/Common Attributes of <group> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	O
resourceID	NP	O
parentID	NP	NP
accessControlPolicyIDs	O	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
Labels	O	O
announceTo	O	O
announcedAttribute	O	O

Table 7.3.13.1-3: Resource Specific Attributes of <group> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	O	NP	m2m:ID	
memberType	M	O	m2m:memberType	Default value is set to 'MIXED'
currentNrOfMembers	NP	NP	xs:integer	No default (This is generated by the hosting CSE and limited by the *maxNrOfMembers* attribute of the <group> resource)
maxNrOfMembers	M	O	xs:integer	No default
memberID	M	O	list ofxs:anyURI	No default
membersAccessControlPolicyIDs	O	O	xs:anyURI	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

Table 7.3.13.1-4: Child resources of <group> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to in Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8
<fanOutPoint>	fanOut	1	Clause 7.3.14

7.3.13.2 <group> resource specific procedure on CRUD operations

This clause describes <group> resource specific procedure on Resource Hosting CSE for CRUD operations.

7.3.13.2.1Create

Primitive specific operation after Recv-C-6.4 "Check validity of resource representation for the given resource type" and before Recv-C-6.5 "Create/Update/Retrieve/Delete/Notify operation is performed ". See clause 7.1.2.2.

1) Primitive specific operation: Validate the provided attributes. It shall also check whether the number of URIs present in the ***memberIDt*** attribute of the group resource representation does not exceed the maximum as specified by the attribute ***maxNrOfMembers***. If the maximum is exceeded, the request shall be rejected with a ***Response Status Code*** indicating "MAX_NUMBER_OF_MEMBER_EXCEEDED" error.
If the ***memberType*** attribute of the <group> resource is not "MIXED", the hosting CSE shall also verify that all the member URIs including sub-groups in the attribute ***memberID*** of the <group> resource representation provided in the request shall conform to the ***memberType*** of the group resource.

2) In the case that the <group> resource contains sub-group member resources, the receiver shall retrieve the ***memberType*** of the 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 "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 subscribe 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.
The ***memberTypeValidated*** attribute shall be set to TRUE if all the members have been validated successfully.

7.3.13.2.2Retrieve

No primitive specific operations.

7.3.13.2.3Update

1) Primitive specific operation 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 7.1.2.2.Primitive specific operation: If the **memberType** attribute of the <group> resource is not "MIXED", the hosting CSE shall verify that all the member URIs including sub-groups in the attribute **memberID** of the <group> resource representation provided in the request shall conform to the **memberType** of the <group> resource.

2) In the case that the <group> resource contains sub-group member resources, the receiver shall retrieve the ***memberType*** of the 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 "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 subscribe 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.
The ***memberTypeValidated*** attribute shall be set to TRUE if all the members have been validated successfully.

3) Primitive specific operation: The hosting CSE shall check whether the number of provided ***memberID*** in the attribute members exceeds the limitation of ***maxNrOfMembers***. If it exceeds, the hosting CSE shall reject the request with ***Response Status Code*** indicating "NOT_ALLOWED". error

7.3.13.2.4 Delete

No primitive specific operations.

7.3.14 Resource Type <fanOutPoint>

7.3.14.1 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 the request is sent to the <fanOutPoint> resource, the request is fanned out to each of the members of the <group> resource indicated by the ***memberID*** 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 TS-0001 Functional Architecture [6].

There is no common attributes, resource specific attributes or xsd file to <fanOutPoint> resource because it's a virtual resource.

7.3.14.2 <fanOutPoint> operations

7.3.14.2.1 Validate the member types

Validate the provided attributes. If the ***memberType*** attribute of the addressed parent resource is not "MIXED", the group hosting CSE may check whether the type of resource to be created is consistent with the addressed parent resource. i.e. if the ***To*** parameter was …/fanOutPoint without any suffix, then the ***memberType*** attribute of the parent group resource determines the type of the addressed resource. Otherwise it is determined by the combination of the ***memberType*** and the child resources addressed in the ***To*** parameter after the fanOutPoint element in the path. If they are not consistent, the request shall be rejected with a ***Response Status Code*** indicating "MEMBER_TYPE_INCONSISTENT" error.

7.3.14.2.2 Sub-group creation for members residing on the same CSE

The group hosting CSE shall obtain URIs of addressed resources from the attribute ***memberID*** 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 perform 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 7.3.13.2.1. The ***To*** parameter of this group Create request may be <memberHosting cseBase>/ <groupHosting remoteCse>/ or <memberHosting cseBase>/ etc. The group hosting CSE shall also provide ***From*** parameter (i.e. group hosting CSE) and sub-group resource representation that contains a ***member***IDattribute with all the members residing on the addressed member Hosting CSE. The sub-group representation may include the attribute ***accessControlPolicyID***s, so that the group hosting CSE has the access right to this sub-group. The ID of the sub-group may be proposed by the group hosting CSE and determined 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.

7.3.14.2.3 Assign URI for aggregation of notification

In the case the created resource is <subscription> resource, the group hosting CSE shall validate if the subscription resource in the received request contains an ***notificationForwardingURI*** attribute. On successful validation, the group hosting CSE shall assign a new ***notificationForwardingURI*** to the attribute for receiving the notifications. The group hosting CSE shall locally maintain the mapping of the new ***notificationForwardingURI*** and the former ***notificationForwardingURI*** if it exists.

7.3.14.2.4 Fanout Request to each member

For each member hosting CSE, the group hosting CSE shall perform the following steps:

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 prefix of primitive parameter ***To*** i.e. <URI of group resource>/fanOutPoint shall be replaced by each URI of member resources derived from the attribute ***memberID*** of the group resource, but excluding the member resources which construct a sub-group. For these members resources contained in a sub-group, the primitive ***To*** of the composed Request shall be <URI of sub-group resource>/fanOutPoint. 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 send 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*** locally stored. 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 locally store the ***Group Request Identifier*** parameter.

7.3.14.3 <fanOutPoint> resource specific procedure on CRUD operations

This sub-clause describes <fanOutPoint> resource specific behaviour for CRUD operations.

7.3.14.3.1 Create

The primitives create the content of all member resources belonging to an existing <group> resource.

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.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 “Check authorization of the Originator": The ***to*** parameter consists of the URI of the group resource plus a suffix marked by /fanOutPoint or /fanOutPoint/.....

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 member types, refer to 7.3.12.2.1
2) Sub-group creation for members residing on the same CSE, refer to 7.3.12.2
3) Assign URI for aggregation of notification, refer to 7.3.12.3
4) Fanout Request to each member, refer to 7.3.12.4
5) The group hosting CSE shall aggregate the Responses after receiving responses from its member resources and sub-groups and aggregate the Responses into a single Response:

Primitive specific operation additional to Recv-6.7 "Create a success response", the Response shall include the aggregated Responses.

7.3.14.4 Retrieve

The primitives retrieve the content of all <member> resources belonging to an existing <group> resource.

Originator:

No primitive specific operations.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 "Check authorization of the Originator": The ***To*** parameter consists of the URI of the <group> resource plus a suffix marked by /fanOutPoint or /fanOutPoint/.....

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) Sub-group creation for members residing on the same CSE, refer to 7.3.12.2
2) Fanout Request to each member, refer to 7.3.12.4
3) The group hosting CSE shall aggregate the Responses after receiving responses from its member resources and sub-groups and aggregate the Responses into a single Response:

Primitive specific operation additional to Recv-6.7 "Create a success response", the Response shall include the aggregated Responses.

7.3.14.4.1 Update

The primitives update the content of all member resources belonging to an existing <group> resource.

Originator:

No primitive specific operations.

Receiver:

Primitive specific operation after Recv-6.2 "Check existence of the addressed resource" and before Recv-6.3 "Check authorization of the Originator": The ***To*** parameter consists of the URI of the <group> resource plus a suffix marked by /fanOutPoint or /fanOutPoint/.....

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 member types ., refer to 7.3.12.1
2) Sub-group creation for members residing on the same CSE , refer to 7.3.13.2
3) Fanout Request to each member. See Clause 7.3.14.2.4
4) The group hosting CSE shall aggregate the Responses after receiving responses from its <member> resources and sub-groups and aggregate the Responses into a single Response:

Primitive specific operation additional to Recv-6.7 "Create a success response", the Response shall include the aggregated Responses.

7.3.14.4.2 Delete

The primitives delete the content of all member resources belonging to an existing <group> resource.

Originator:

No primitive specific operations.

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 marked by /fanOutPoint or /fanOutPoint/.....

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 member types , refer to 7.3.12.1
2) Sub-group creation for members residing on the same CSE , refer to 7.3.14.2.2
3) Fanout Request to each member. See Clause 7.3.14.2.4
4) The group hosting CSE shall aggregate the Responses after receiving responses from its <member> resources and sub-groups and aggregate the Responses into a single Response:

Primitive specific operation additional to Recv-6.7 "Create a success response", the Response shall include the aggregated Responses.

7.3.15 Resource Type <mgmtObj>

7.3.15.1 Introduction

The <mgmtObj> resource contains management data which represents individual M2M management functions. It represents a general structure to map to external management technology 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 7.3.15.1-1: Universal/Common Attributes of <mgmtObj> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	O
resourceID	NP	O
parentID	NP	NP
accessControlPolicyIDs	O	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.15.1-2: Resource Specific Attributes of <mgmtObj> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	O	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
mgmtLink	O	O	m2m:mgmtLinkRef	No default

Table 7.3.15.1-3: Child resources of <mgmtObj> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.15.2 <mgmtObj> resource specific procedure on CRUD operations

This clause describes <mgmtObj> resource specific procedure on resource Hosting CSE for CRUD operations.

The procedures are defined for management when external management technologies are used. When service layer management is performed, generic procedures defined in clause 7.1.2 shall comply for resource creation, update, retrieval and deletion. Procedures additional to resource manipulations to perform the management are further defined in Annex D.

7.3.15.2.1Create

Primitive specific operation before Orig-C-1.0 “Compose Request primitive”:

1) Primitive specific operation: If the originator is the managed entity, it shall generate the <mgmtObj> resource representation based on the external management object information of the managed entity to be exposed. The ***objectIDs*** and ***objectPaths*** attributes may be set with the Request.

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” if the originator is an IN-AE:

1) "Identify the managed entity and the management protocol ".

Primitive specific operation: the receiver shall generate the external management 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 external management object shall be added based on the ***objectIDs*** and ***objectPaths*** provided in the request primitive and the protocol specific data model being used. The receiver may also choose to let the managed entity decide the target location where the generated external management object shall be added using protocol specific mechanism.

1) "Establish a management session with the managed entity".

2) "Send the management request(s) to the managed entity corresponding to the received Request primitive ". If the receiver receives an error response from the managed entity because the external management object to be added already exists on the managed entity, the receiver shall check (by using e.g. OMA-DM Get command or TR069 GetParameterValues/GetParameterAttributes command) if the existing external management 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 “ALREADY_EXISTS” error in the Response primitive. The receiver shall also record the location where the external management object is added to the managed entity in the successful case. The ***objectIDs*** and ***objectPaths*** attributes may be set with the Request.

3) The receiver may repeat Step 4 in order to add to the managed entity the external management objects that are mapped from the mandatory sub-resources (including any descendants) that are required to be created automatically with the default attribute values.

7.3.15.2.2 Retrieve

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” if the originator is an IN-AE:

1) "Identify the managed entity and the management protocol".

2) "Locate the external management objects to be managed on the managed entity".

3) "Establish a management session with the managed entity".

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 external management object information if required according to the local policy.

7.3.15.2.3 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 which without any content as the payload part of the primitive by addressing specific attribute to start the management procedure.

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” if the originator is IN-AE.

1) "Identify the managed entity and the management protocol".

2) "Locate the external management objects to be managed on the managed entity".

3) "Establish a management session with the managed entity".

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 external management object information if required according to the local policy.

7.3.15.2.4 Delete

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” if the originator is IN-AE.

1) "Identify the managed entity and the management protocol".

2) "Locate the external management objects to be managed on the managed entity".

3) "Establish a management session with the managed entity".

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 external management object information if required according to the local policy.

7.3.16 Resource Type <mgmtCmd>

7.3.16.1 Introduction

The <mgmtCmd> resource shall contain the following attributes and child resource as illustrated in Table 7.3.16.1-2, Table 7.3.16.1-3, and Table 7.3.16.1-4. The data type and default value of these attributes and child resources are included in the tables.

Table 7.3.16.1-1: Data type definition of <mgmtCmd> resource

Data Type ID	File Name	Note
mgmtCmd	CDT-mgmtCmd-v1_0_0.xsd	

Table 7.3.16.1-2: Universal/Common Attributes of <mgmtCmd> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.16.1-3: 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	O	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	M	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 7.3.16.1-4: Child resources of <mgmtCmd> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	clause 7.3.8
<execInstance>	[variable]	1	clause 7.3.17

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.

7.3.16.2 <mgmtCmd> resource specific procedure on CRUD oerations

This clause describes <mgmtCmd> resource specific procedures for CRUD operations.

7.3.16.2.1Create

This procedure shall use the Create common operations detailed in clause 7.1.2.1 without primitive specific actions. The Originator shall use the steps Orig-C-1.0, Orig-C-2.0, and Orig-C-3.0 as described in clause 7.1.2.1. The Receiver shall use the steps Rcv-C-1.0 to Rcv-C-11.0 as described in clause 7.1.2.1.

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.

7.3.16.2.2Retrieve

This procedure shall use the Retrieve common operations detailed in clause 7.2 without primitive specific actions. The Originator shall use the steps Orig-R-1.0, Orig-R-2.0, and Orig-R-3.0 as described in clause 7.2. The Receiver shall use the steps Rcv-R-1.0 to Rcv-R-9.0 as described in clause 7.2.

7.3.16.2.3Update

7.3.15.2.3.1 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 7.2, without primitive specific actions.

The Originator shall use the steps Orig-U-1.0, Orig-U-2.0, and Orig-U-3.0 as described in clause 7.2. The Receiver shall use the steps Rcv-U-1.0 to Rcv-U-11.0 as described in clause 7.2.

If the Originator attempts to update attributes ***resourceType***, ***resourceID*** or ***cmdType***, the Receiver shall generate a ***Response Status Code*** indicating "NO_PRIVILEGE" error..

If the Originator attempts to update attributes ***execTarget***, ***execMode***, but the <mgmtCmd> has child resource <execInstance> already created, the Receiver shall generate a ***Response Status Code*** indicating "CONTENTS_UNACCEPTABLE" error.

If the Originator attempts to update attributes any attribute with a value which is not allowed, the Receiver shall generate a ***Response Status Code*** indicating "CONTENTS_UNACCEPTABLE" error.

If the Update primitive for <mgmtCmd> does address the ***execEnable*** attribute of the <mgmtCmd>, it effectively triggers an Execute <mgmtCmd> procedure, see clause 7.3.15.2.3.2.

7.3.15.2.3.2 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 with the following primitive specific operation after Rcv-U-4.0 and before Rcv-U-5.0:

1) The Receiver shall identify the managed entity and the management protocol. The ***execTarget*** attribute of <mgmtCmd> indicates the managed entity.

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.

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 as INITIATED. The Receiver shall set the execTarget attribute of each <execInstance> sub-resource to the URI of each target <node> resource.

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***).

The Receiver shall establish a management session with the identified managed entity.

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.

After receiving completion response from the managed entity, the Receiver shall set ***execStatus*** attribute of corresponding <execInstance> to FINISHED.

If the Update primitive for <mgmtCmd> does not address the ***execEnable*** attribute of the <mgmtCmd>, it effectively triggers an Update <mgmtCmd> procedure, see clause 7.3.16.2.3.

7.3.16.2.4Delete

This procedure is based on the Delete common operations detailed in clause 7.2.

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 un-successful 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" error 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.

7.3.17 Resource Type <execInstance>

7.3.17.1 Introduction

The <execInstance> resource shall contain the following child resource and attributes.

Table 7.3.17.1-1: Data type definition of <execInstance> resource

Data Type ID	File Name	Note
execInstance	CDT-execInstance-v1_0_0.xsd	

Table 7.3.17.1-2: Universal/Common Attributes of <execInstance> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicyIDs	O	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.17.1-3: Resource Specific Attributes of <execInstance> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
execStatus	NP	O	m2m:execStatusType	INITIATED, PENDING, FINISHED, CANCELLING, CANCELLED STATUS_NON_CANCELLABLEL Default=INITIATED
execResult	NP	O	xs:execResultType	No default
execDisable	NP	O	xs:boolean	No default
execTarget	O	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
execReqArgs	O	O	m2m:execReqArgsListType	No default

Table 7.3.17.1-4: Child Resources of <execInstance> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.17.2 <execInstance> resource specific procedure on CRUD oerations

This clause describes <execInstance> resource specific procedures for CRUD operations.

7.3.17.2.1Update (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 7.2.

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" error 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" error. 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" error, and the ***execStatus*** attribute is changed to STATUS_NON_CANCELLABLE.

7.3.17.2.2Retrieve

This procedure shall use the Retrieve common operations detailed in clause 7.2, without primitive specific actions. The Originator shall use the steps Orig-R-1.0, Orig-R-2.0, and Orig-R-3.0 as described in clause 7.1.2.1. The Receiver shall use the steps Rcv-R-1.0 to Rcv-R-9.0 as described in clause 7.2.

7.3.17.2.3Delete

This procedure is based on the Delete common operations detailed in clause 7.2.

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" error. 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.

7.3.18 Resource Type <node>

7.3.18.1 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 7.3.18.1-1: Data type definition of <node> resource

Data Type ID	File Name	Note
node	CDT-node-v1_0_0.xsd	

Table 7.3.18.1-2: Universal/Common Attributes of <node> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	O	NP
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.18.1-3: 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	NP	m2m:ID	

Table 7.3.18.1-4: Child resources of <node> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<mgmtObj>	[variable]	0..n	7.3.15, See Annex D
<subscription>	[variable]	0..n	7.3.8

7.3.18.2 <node> resource specific procedure on CRUD operations

7.3.18.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.18.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.18.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2

7.3.18.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.19 Resource Type <m2mServiceSubscriptionProfile>

7.3.19.1 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 TS-0001 Functional Architecture [6].

Table 7.3.19.1-1: Data type definition of <m2mServiceSubscriptionProfile> resource

Data Type ID	File Name	Note
m2mServiceSubscriptionProfile	CDT-m2mServiceSubscriptionProfile-V1_0_0.xsd	

Table 7.3.19.1-2: Universal/Common Attributes of <m2mServiceSubscriptionProfile>

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicy IDs	O	O
creationTime	NP	NP
labels	O	O
lastModifiedTime	NP	NP

Table 7.3.19.1-3: Resource Specific Attributes of <m2mServiceSubscriptionProfile>

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
serviceRoles	O	O	m2m:serviceRoles	

Table 7.3.19.1-4: Child resources of <m2mServiceSubscriptionProfile>

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8
<serviceSubscribedNode >	[variable]	0..n	Clause 7.3.20

7.3.19.2 <m2mServiceSubscriptionProfile> resource specific procedure on CRUD operations

This clause describes <m2mServiceSubscriptionProfile> resource specific behaviour for CRUD operations.

7.3.19.2.1Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.19.2.2Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.19.2.3Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.19.2.4Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.20 Resource Type <serviceSubscribedNode>

7.3.20.1 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 TS-0001 Functional Architecture [6].

Table 7.3.20.1-1: Data type definition of <serviceSubscribedNode> resource

Data Type ID	File Name	Note
serviceSubscribedNode	CDT-serviceSubscribedNode-V1_0_0.xsd	

Table 7.3.20.1-2: Universal/Common Attributes of <serviceSubscribedNode> resource

Attribute Name	Request Optionality	
	Create	Update
@resourceName	NP	NP
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
expirationTime	O	O
accessControlPolicy IDs	O	O
creationTime	NP	O
labels	O	O
lastModifiedTime	NP	NP

Table 7.3.20.1-3: Resource Specific Attributes of <serviceSubscribedNode> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
nodeID	M	O	m2m:nodeID	
CSE-ID	O	O	m2m:ID	
deviceIdentifier	O	O	list of m2m:deviceID	
ruleLinks	O	O	list of xs:anyURI	

Table 7.3.20.1-4: Child resources of <serviceSubscribedNode> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	7.3.8

7.3.20.2 <serviceSubscribedNode> resource specific procedure on CRUD operations

This clause describes <serviceSubscribedNode> resource specific behaviour for CRUD operations.

7.3.20.2.1Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.20.2.2Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.20.2.3Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.20.2.4Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.21 Resource Type <pollingChannel>

7.3.21.1 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 TS-0001 Functional Architecture [6].

Table 7.3.21.1-1: Data type definition of <pollingChannel> resource

Data Type ID	File Name	Note
pollingChannel	CDT-pollingChannel-V1_0_0.xsd	

Table 7.3.21.1-2: Universal/Common Attributes of <pollingChannel> 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
expirationTime	O	O
accessControlPolicyIDs	O	NP
labels	O	O

Table 7.3.21.1-3: Child resources of <pollingChannel> resource

Child Resource Type	Name	Multiplicity	Ref. to Resource Type Definition
<pollingChannelURI>	pollingChannelURI	1	Clause 7.3.22

7.3.21.2 <pollingChannel> resource specific procedure on CRUD operations

This clause describes <pollingChannel> resource specific behaviour for CRUD operations.

7.3.21.2.1Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

Same as the generic procedures in clause 7.1.2.2 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 is failed, then the Hosting CSE shall return response primitive with a ***Response Status Code*** indicating "NO_PRIVILEGE" error.

7.3.21.2.2Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.21.2.3Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.21.2.4Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.22 Resource Type <pollingChannelURI>

7.3.22.1 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 TS-0001 Functional Architecture [6].

There is no data type definition for <pollingChannelURI> resource because it's a virtual resource type.

7.3.22.2 <pollingChannelURI> resource specific procedure on CRUD operations

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.

7.3.22.2.1Create

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.

7.3.22.2.2Retrieve

Originator: shall execute Originator actions in clause 7.1.2.1 as a service layer long polling request. It's 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 TS-0001 Functional Architecture [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 7.1.2.2:

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 "ACCESS_DENIED" 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 pending request(s).

Else

Wait for a request for the Originator until t he ***Request Expiration Timestamp*** of the Originator's request. If a request is available before the ***Request Expiration Timestamp*** timeout, create a Response primitive including Pending Requests primitive parameter. Otherwise, create a response primitive with a ***Response Status Code*** indicating "REQUEST_TIMEOUT" error..

7.3.22.2.3 Update

The present document does not define Update operation on a <pollingChannelURI> resource. An Update request for the resource shall be rejected.

7.3.22.2.4 Delete

The present document does not define Delete operation on a <pollingChannelURI> resource. A Delete request for the resource shall be rejected.

7.3.23 Resource Type <statsConfig>

7.3.23.1 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 9.6.22 and 10.2.15 of TS-0001 Functional Architecture [6].

Table 7.3.23.1-1: Data type definition of <statsConfig>

Data Type ID	File Name	Note
statsConfig	*CDT-statsConfig-v1_0_0.xsd*	

Table 7.3.23.1-2: Universal/Common Attributes of <stateConfig> resource

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	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.23.1-3: Resource Specific Attributes of <stateConfig> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	NP	NP	m2m:ID	

Table 7.3.23.1-4: Child resources of <statsConfig> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<eventConfig>	[variable]	0..n	7.3.24
<subscription>	[variable]	0..n	7.3.8

7.3.23.2 <statsConfig> resource-specific procedure on CRUD operations

7.3.23.2.1 Create

Originator:

No change from the generic procedure in clause 7.1.2.1

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.1.2.1 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.

7.3.23.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.23.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.23.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.24 Resource Type <eventConfig>

7.3.24.1 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 9.6.23 and 10.2.15 of TS-0001 Functional Architecture [6].

Table 7.3.24.1-1: Data type definition of <eventConfig>

Data Type ID	File Name	Note
eventConfig	CDT-eventConfig-v1_0_0.xsd	

Table 7.3.24.1-2: Universal/Common Attributes of <eventConfig> resource

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	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.24.1-3: Resource Specific Attributes of <eventConfig> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	NP	NP	m2m:ID	No default
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)

Table 7.3.24.1-4: Child Resources of <eventConfig> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.24.2 <eventConfig> resource-specific procedure on CRUD operations

7.3.24.2.1 Create

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.1.2.1, 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 7.3.24.1-3 for value restrictions and default settings pertaining to the attributes of <eventConfig>.

Receiver:

No change from the generic procedure in clause 7.1.2.2.

7.3.24.2.2 Retrieve

Originator:

No change from the generic procedure in clause 7.1.2.1.

Receiver:

No change from the generic procedure in clause 7.1.2.2.

7.3.24.2.3 Update

Originator:

No change from the generic procedure in clause 7.1.2.1.

Receiver:

No change from the generic procedure in clause 7.1.2.2.

7.3.24.2.4 Delete

Originator:

No change from the generic procedure in clause 7.1.2.1.

Receiver:

No change from the generic procedure in clause 7.1.2.2.

7.3.25 Resource Type <statsCollect>

7.3.25.1 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 9.6.24 and 10.2.15 of TS-0001 Functional Architecture [6].

Table 7.3.25.1-1: Data type definition of <statsCollect>

Data Type ID	File Name	Note
statsCollect	CDT-statsCollect-v1_0_0.xsd	

Table 7.3.25.1-2: Universal/Common Attributes of <statsCollect> resource

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	O	O
lastModifiedTime	NP	NP
labels	O	O

Table 7.3.25.1-3: Resource Specific Attributes of <statsCollect> resource

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
creator	NP	NP	m2m:ID	No default
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	M	NP	m2m:ID	Unique ID of entity (e.g., AE, CSE) for which statistics will be collected No default
statsRuleStatus	M	O	m2m:statsRuleStatusType	ACTIVE INACTIVE No default
statModel	M	O	m2m:statModelType	EVENTBASED Default=EVENTBASED
collectPeriod	O	O	m2m:scheduleEntries	No default (see Table 7.3.9.1-3 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 7.3.25.1-4: Child Resources of <statsCollect> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.25.2 <statsCollect> resource-specific procedure on CRUD operations

7.3.25.2.1 Create

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.1.2.1, 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 7.3.25.1-3 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 7.1.2.2, 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 operation 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.

7.3.25.2.2 Retrieve

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.1.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.1.2.2.

7.3.25.2.3 Update

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.1.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.1.2.2, 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.

7.3.25.2.4 Delete

Originator:

This procedure follows the Generic Resource Request Procedure for Originator specified in clause 7.1.2.1.

Receiver:

This procedure follows the Generic Request Procedure for Receiver specified in clause 7.1.2.1.

7.3.26 Announced resource type

7.3.26.1 Introduction

A resource can be announced to one or more remote CSEs to inform the remote CSEs of the existence of the original resource. An announced resource can have a limited set of attributes and a limited set of child resources from the original resource. The announced resource includes a link to the original resource hosted by the original resource-hosting CSE.

All announced resources have the same procedures regardless of the announced resource types.

Table 7.3.26.1-1: Data type definition of announced Resource

Data Type ID	File Name	Note
Actual Data Type ID	CDT-accessControlPolicy-v1_0_0.xsd CDT-remoteCSE-v1_0_0.xsd CDT-AE-v1_0_0.xsd CDT-container-v1_0_0.xsd CDT-contentInstance-v1_0_0.xsd CDT-schedule-v1_0_0.xsd CDT-locationPolicy-v1_0_0.xsd CDT-group-v1_0_0.xsd CDT-node-V1_0_0.xsd	

Table 7.3.26.1-2: Universal/Common Attributes of announcedResource

Attribute Name	Request Optionality	
	Create	Update
resourceType	NP	NP
resourceID	NP	NP
parentID	NP	NP
accessControlPolicy IDs	O	O
creationTime	NP	NP
expirationTime	O	O
lastModifiedTime	NP	NP
labels	O	O
link	M	O

Each announced resource type has the resource specific attributes that is the subset of the original resource.

Table 7.3.26.1-3: 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	the same data type defined at 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	the same data type defined at the original resource	this attribute shall be set to the same value with the attribute at the original resource

7.3.26.2 Resource specific procedure on CRUD operations

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 7.2.3.9 and clause 7.1.2.2.

7.3.26.2.1 Create

.

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.26.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

In case of the ***Result Content*** information is set to the "original-resource", the Rcv-R-6.5 shall be changed as follows:

Rcv-R-6.5"Read the original resource whose address is provided by the ***link*** attribute of the announced resource"

7.3.26.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.26.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.27 Resource Type latest

7.3.27.1 Introduction

The <latest> resource is a virtual resource because it does not have a representation. It is a child resource of the <container> resource. Whenever a request addresses the <latest> resource, the Hosting CSE shall apply the request to the latest <contentInstance> resource among all existing <contentInstance> resources of the <container> resource.

7.3.27.2 <latest> Resource Specific Procedure on CRUD Operations

This sub-clause describes <latest> resource specific behaviour for operations. Among operations, only Retrieve and Delete operations shall be allowed for the <latest> resource.

7.3.27.2.1 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 "OPERATION_NOT_ALLOWED" error.

   l) "Send the Response primitive".

7.3.27.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1 except the following modification:

Recv-6.2 Check the existence of the latest <contentInstance> resource among all existing <contentInstance> resources in the parent <container> 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.

7.3.27.2.3 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 ”OPERATION_NOT_ALLOWED” error.

   m) "Send the Response primitive".

7.3.27.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1 except the following modification:

Recv-6.2 Check the existence of the latest <contentInstance> resource among all existing <contentInstance> resources in the parent <container> 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.

7.3.28 Resource Type oldest

7.3.28.1 Introduction

The <oldest> resource is a virtual resource because it does not have a representation. It is a child resource of the <container> resource. Whenever a request addresses the <oldest> resource, the Hosting CSE shall apply the request to the oldest <contentInstance> resource among all existing <contentInstance> resources of the <container> resource.

7.3.28.2 <oldest> Resource Specific Procedure on CRUD Operations

Among operations, only Retrieve and Delete operations shall be allowed for the <oldest> resource.

7.3.28.2.1 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.

   n) "Send the Response primitive".

7.3.28.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1 except the following modification:

Recv-6.2 Check the existence of the oldest <contentInstance> resource among all existing <contentInstance> resources in the parent <container> 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.

7.3.28.2.3 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 "OPERATION_NOT_ALLOWED" error.

   o) "Send the Response primitive".

7.3.28.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.1 except the following modification:

Recv-6.2 Check the existence of the oldest <contentInstance> resource among all existing <contentInstance> resources in the parent <container> 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.

7.3.29 Resource Type <serviceSubscribedAppRule>

7.3.29.1 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 TS-0001 Functional Architecture [6].

Table 7.3.29.1-1: Data type definition of <serviceSubscribedAppRule> resource

Data Type ID	File Name	Note
serviceSubscribedAppRule	CDT-serviceSubscribedAppRule-v1_0_0.xsd	

Table 7.3.29.1-2: Universal/Common Attributes of <serviceSubscribedAppRule> resource

Attribute Name	Request Optionality		Resource Specific Note
	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	

Table 7.3.29.1-3: 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	

Table 7.3.29.1-4: Child resources of <serviceSubscribedAppRule> resource

Child Resource Type	Child Resource Name	Multiplicity	Ref. to Resource Type Definition
<subscription>	[variable]	0..n	Clause 7.3.8

7.3.29.2 <serviceSubscribedAppRule> resource specific procedure on CRUD operations

This clause describes <serviceSubscribedAppRule> resource specific primitive behaviour for CRUD operations.

7.3.29.2.1 Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.29.2.2 Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.29.2.3 Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.3.29.2.4 Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

7.4 Primitive-specific procedures and definitions

7.4.1 Notification data object and procedures

7.4.1.1 Notification data object

Notification procedures represent a special case of the generic procedures defined in clause 7.1.2, 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 Functional Architecture [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.

Table 7.4.1.1-1: Data Type Definition of notification data object

Data Type ID	File Name	Note
notification	*CDT-notification-v1_0_0.xsd*	

Table 7.4.1-2: Data Types for notification parameters

Parameter Name	Request Optionality	Data Type	Default Value and Constraints
	N		
notification	O	m2m:notificaiton	
aggregatedNotification	O	m2m:aggregatedNotification	
responsePrimitive	O	m2m:responsePrimitive	

7.4.1.2 Notification procedures

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.

The following sub-clauses specify the notification procedures for each of the above use cases.

7.4.1.2.1 Notification for modification of subscribed 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. In that case, go to the step 2.0. Otherwise, the Originator shall discard the corresponding event
- If the *eventNotificationCriteria* attribute is not configured, 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 2.1to 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 'modifiedAttribute', 'wholeResource' or optionally 'referenceOnly'

- If the value of *notificationContentType* is set to 'modifiedAttribute', the Notify request primitive shall include the modified attribute(s) only
- If the value of *notificationContentType* is set to 'wholeResource', the Notify request primitive shall include the whole subscribed-to resource
- If the value of *notificationContentType* is set to 'referenceOnly', the Notify request primitive shall include an URI of a corresponding <subscription> resource
- If the *notificationContentType* attribute is not configured, the default value is set to 'wholeResource'

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 next step
- If the *notificationEventCat* attribute is not configured,then continue with other step

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. Then continue with other step

NOTE: The use of some attributes such as *rateLimit*, *batchNotify* and *preSubscriptionNotify* is not supported in this release of the document.

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 periodsgiven in the 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 after the reachability recovery (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

Originator: After reachability recovery, 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.

7.4.1.2.2 Subscription Verification during Subscription Creation

Originator*:*

When the Originator is triggered to perform subscription verification (clause 7.3.8.2.1) 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 *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.

7.4.1.2.3 Notification for Subscription Deletion

Originator*:*

When the <subscription> resource is deleted, the Originator shall send a Notify request primitive with subscriptionDeletion element of the notification data object set as TRUE and *subscriptionReference* element set as the URI of the <subscription> resource.

7.4.1.2.4 Notification for Asynchronous Non-blocking Request

Originator*:*

When the requested operation for a nonBlockingAsynch 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.

1) The Originator shall compose a Request primitive with following parameter settings:
    a. The ***From*** parameter shall be set to the ID of the Originator (i.e. hosting CSE which hosts the targeted resource in the previously received nonBlockingAsynch request).
    b. The ***To*** parameter shall be set to the Originator of the previsouly received nonBlockingAsynch request if no notification target is provided in the ***Response Type*** parameter or to the notificationURI in the ***Response Type*** parameter.

c. The ***Response Type*** : This parameter shall be set to either nonBlockingSynch or nonBlockingAsynch. If the Originator selects to send the Notification in nonBlockingAsynch mode, the Originator shall include empty notification target.

d. The ***Content*** parameter shall be set to the response to the previously received nonBlockingAsynch request as m2m:responsePrimitive.

2) The Originator shall send the Request primitive. See clause 7.2.1.2 for detail.

Receiver*:*

No change from the generic procedure in clause 7.1.2.2.

7.4.1.2.5 Notification for subscription via group

Whenever the subscribed to resources' modification triggers a notification procedure as definedin clause 7.4.1.2.1 and the subscription relationship is established through group resource, the following procedure shall be performed.

The **Member hosting CSE** shall perform the steps defined in 7.4.1.2.1.

The **Group hosting CSE** shall perform the following steps in order**:**

1) Validate if the notification is sent from 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 "ACCESS_DENIED" 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 obeject. The group hosting CSE shall aggregate the notification requests into an aggregatedNotification element of the notification data object. The timing of aggregation is done as per the group hosting CSE's local policy which is out of scope of this specification.

3) Send the aggregated notification to the *notificationURI* according to the *notificationForwardingURI* element in the notification data object. 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 may also stop aggregating notification requests depending on its own policy. The group hosting CSE shall not stop aggregating notification requests before the corresponding subscription expires.

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.

7.4.2 Elements contained in the primitive Content

Clauses 7.1.1.1 and 7.1.1.2 enumerate the forms that the primitive Content parameter takes in various Request and Response cases. 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 7.4.2-1: Elements used for request content

Element Name	Applicable Operations	Data Type	Defined in
m2m:notification	N	m2m:notification	*CDT-notification-v1_0_0.xsd*
m2m:aggregatedNotification	N	m2m:aggregatedNotification	*CDT-notification-v1_0_0.xsd*
m2m:attributeList	R	m2m:attributeList	*CDT-requestPrimitive-v1_0_0.xsd*
m2m:responsePrimitive	N	m2m:responsePrimitive	*CDT-responsePrimitive-v1_0_0.xsd*

The following elements are defined for use in the content parameter of a response:

Table 7.4.2-2: Elements used for response content

Element Name	Applicable Operations	Data Type	Element is Defined in
m2m:resource	C R U	m2m:resourceWrapper	*CDT-responsePrimitive-v1_0_0.xsd*
m2m:URIList	R	m2m:listOfURIs	*CDT-responsePrimitive-v1_0_0.xsd*
m2m:aggregatedResponse	C R U D	m2m:aggregatedResponse	*CDT-responsePrimitive-v1_0_0.xsd*

8 Representation of primitives in data transfer

8.1 Introduction

This clause defines the representation of request and response primitives as XML documents or JSON texts. 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 it two places:

1) It can be used when transmitting primitives over communication protocols such as HTTP, CoAP or MQTT. 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 or XML 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 and JSON 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 and JSON representations require the explicit encoding of the names of primitive parameters, resource attributes, (in the case of XML) resource types and complex data types 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 that follow.

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	Occurs in	Short Name
Operation	Request	***op***
To	Request, Response	***to***
From	Request, Response	***fr***
Request Identifier	Request, Response	***rqi***
Resource Type	Request	***ty***
Name	Request	***nm***
Content	Request, Response	***pc***
Originating Timestamp	Request, Response	***ot***
Request Expiration Timestamp	Request	***rqet***
Result Expiration Timestamp	Request, Response	***rset***
Operation Execution Time	Request	***oet***
Response Type	Request	***rt***
Result Persistence	Request	***rp***
Result Content	Request	***rcn***
Event Category	Request, Response	***ec***
Delivery Aggregation	Request	***da***
Group Request Identifier	Request	***gid***
Filter Criteria	Request	***fc***
createdBefore	Request	***crb***
createdAfter	Request	***cra***
modifiedSince	Request	***ms***
unmodifiedSince	Request	***us***
stateTagSmaller	Request	***sts***
stateTagBigger	Request	***stb***
expireBefore	Request	***exb***
expireAfter	Request	***exa***
labels	Request	***lbl***
resourceType	Request	***rty***
sizeAbove	Request	***sza***
sizeBelow	Request	***szb***
contentType	Request	***cty***
limit	Request	***lim***
attribute	Request	***atr***
filterUsage	Request	***fu***
Discovery Result Type	Request	***drt***
Response Status Code	Response	***rsc***

XML serialized representations of primitives employ root element names to differentiate between request and response primitive types (see clause 8.3). These root element names shall be translated into short names as in Table 8.2.2-2.

Table 8.2.2-2: Primitive root element short names

Root Element Name	Occurs in	Short Name
requestPrimitive	Request	***req***
responsePrimitive	Response	***rsp***

8.2.3 Resource attributes

In protocol bindings resource attributes names shall be translated into short names of Table 8.3-1.

Table 8.2.3-1: Resource attribute short names (1/5)

Attribute Name	Occurs in	Short Name
accessControlPolicyIDs	All except accessControlPolicy, contentInstance	***acpi***
announcedAttribute	accessControlPolicy, AE, container, contentInstance, group, locationPolicy, mgmtObj, node, remoteCSE, schedule	***aa***
announceTo	accessControlPolicy, AE, container, contentInstance, group, locationPolicy, mgmtObj, node, remoteCSE, schedule	***at***
creationTime	All	***ct***
expirationTime	All except contentInstance, CSEBase	***et***
lastModifiedTime	All	***lt***
parentID	All except CSEBase	***pi***
resourceID	All	***ri***
stateTag	container, contentInstance, delivery, request	***st***
resourceName	All	***rn***
privileges	accessControlPolicy	***pv***
selfPrivileges	accessControlPolicy	***pvs***
App-ID	AE	***api***
AE-ID	AE	***aei***
appName	AE	***apn***
pointOfAccess	AE, CSEBase, remoteCSE	***poa***
ontologyRef	AE, container, contentInstance	***or***
nodeLink	AE, CSEBase, remoteCSE	***nl***
creator	container, contentInstance,eventConfig, group, pollingChannel, statsCollect, statsConfig, subscription	***cr***
maxNrOfInstances	container	***mni***
maxByteSize	container	***mbs***
maxInstanceAge	container	***mia***
currentNrOfInstances	container	***cni***

Table 8.2.3-2: Resource attribute short names (2/5)

Attribute Name	Occurs in	Short Name
currentByteSize	container	***cbs***
latest	container	***la***
locationID	container	***li***
contentInfo	contentInstance	***cnf***
contentSize	contentInstance	***cs***
content	contentInstance	***con***
cseType	CSEBase, remoteCSE	***cst***
CSE-ID	CSEBase, remoteCSE, service SubscribedNode	***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***
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***
currentNrOfMembers	group	***cnm***
maxNrOfMembers	group	***mnm***
memberID	group	***mid***
membersAccessControlPolicyIDs	group	***macp***
memberTypeValidated	group	***mtv***
consistencyStrategy	group	***csy***
groupName	group, subscription	***gn***
locationSource	locationPolicy	***los***
locationUpdatePeriod	locationPolicy	***lou***
locationTargetId	locationPolicy	***lot***
locationServer	locationPolicy	***lor***
locationContainerID	locationPolicy	***loi***
locationContainerName	locationPolicy	***lon***
locationStatus	locationPolicy	***lost***
serviceRoles	m2mServiceSubscriptionProfile	***svr***
description	mgmtCmd, mgmtObj, all management resources from firmware	***dc***
cmdType	mgmtCmd	***cmt***
mgmtDefinition	mgmtObj, all management resources from firmware	***mgd***
objectIDs	mgmtObj	***obis***

Table 8.2.3-3: Resource attribute short names (3/5)

Attribute Name	Occurs in	Short Name
objectPaths	mgmtObj	***obps***
nodeID	node	***ni***
hostedCSELink	node	***hcl***
CSEBase	remoteCSE	***cb***
M2M-Ext-ID	remoteCSE	***mei***
Trigger-Recipient-ID	remoteCSE	***tri***
requestReachability	remoteCSE	***rr***
originator	request	***og***
metaInformation	request	***mi***
requestStatus	request	***rs***
operationResult	request	***ol***
operation	request	***opn***
requestID	request	***rid***
scheduleElement	schedule	***se***
deviceIdentifier	serviceSubscribedNode	***di***
statsCollectID	statsCollect	***sci***
collectingEntityID	statsCollect	***cei***
collectedEntityID	statsCollect	***cdi***
status	areaNwkDeviceInfo	***ss***
statsRuleStatus	statsCollect	***srs***
statModel	statsCollect	***sm***
collectPeriod	statsCollect	***cp***
eventNotificationCriteria	subscription	***enc***
expirationCounter	subscription	***exc***
notificationURI	subscription	***nu***
notificationForwardingURI	subscription	***nfu***
batchNotify	subscription	***bn***
rateLimit	subscription	***rl***
preSubscriptionNotify	subscription	***psn***
pendingNotification	subscription	***pn***
notificationStoragePriority	subscription	***nsp***
latestNotify	subscription	***ln***
notificationContentType	subscription	***nct***
notificationEventCat	subscription	***nec***
subscriberURI	subscription	***su***
version	firmware, software	***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***
activateStatus	software, areaNwkInfo	***acts***
memAvailable	memory	***mma***
memTotal	memory	***mmt***

Table 8.2.3-4: Resource attribute short names (4/5)

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***
capabilityName	deviceCapability	***can***
attached	deviceCapability	***att***
capabilityActionStatus	deviceCapability	***cas***
enable	deviceCapability	***ena***
disable	deviceCapability	***dis***
currentState	deviceCapability	***cus***
reboot	reboot	***rbo***
factoryReset	reboot	***far***
logTypeId	eventLog	***lgt***
logData	eventLog	***lgd***
logActionStatus	eventLog	***lgs***
logStart	eventLog	***lga***
logStop	eventLog	***lgo***
name	cmdhPolicy, firmware, software	***nam***
mgmtLink	cmdhPolicy, activeCmdhPolicy, cmdhDefaults, cmdhNetworkAccessRules, cmdhNwAccessRule	***cmlk***
order	cmdhDefEcValue, cmdhLimits	***od***
defEcValue	cmdhDefEcValue	***dev***
requestOrigin	cmdhDefEcValue, cmdhLimits	***ror***
requestContext	cmdhDefEcValue, cmdhLimits	***rct***
requestContextNotification	cmdhDefEcValue, cmdhLimits	***rcn***
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 (5/5)

Attribute Name	Occurs in	Short Name
minReqVolume	cmdhNwAccessRule	***mrv***
backOffParameters	cmdhNwAccessRule	***bop***
otherConditions	cmdhNwAccessRule	***ohc***
maxBufferSize	cmdhBuffer	***mbfs***
storagePriority	cmdhBuffer	***sgp***
applicableCredIDs	serviceSubscribedAppRule	***aci***
allowedApp-IDs	serviceSubscribedAppRule	***aai***
allowedAEs	serviceSubscribedAppRule	***aae***
NOTE: marked short names have been already assigned in primitive 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	***csb***
delivery	***dlv***
eventConfig	***evcg***
execInstance	***exin***
fanOutPoint	***fopt***
group	***grp***
groupAnnc	***grpA***
locationPolicy	***lcp***
locationPolicyAnnc	***lcpA***
m2mServiceSubscriptionProfile	***mssp***
mgmtCmd	***mgc***
mgmtObj	***mgo***
mgmtObjAnnc	***mgoA***
node	***nod***
nodeAnnc	***nodA***
pollingChannel	***pch***
pollingChannelURI	***pcu***
remoteCSE	***csr***
remoteCSEAnnc	***csrA***
request	***req***
schedule	***sch***
scheduleAnnc	***schA***
serviceSubscribedAppRule	***asar***
serviceSubscribedNode	***svsn***
statsCollect	***stcl***
statsConfig	***stcg***
subscription	***sub***
firmware	***fwr***
software	***swr***
memory	***mem***
areaNwkInfo	***ani***
areaNwkDeviceInfo	***andi***
battery	***bat***
deviceInfo	***dvi***
deviceCapability	***dvc***
reboot	***rbt***
eventLog	***evl***
cmdhPolicy	***cmp***
activeCmdhPolicy	***acmp***
cmdhDefaults	***cmdf***
cmdhDefEcValue	***cmdv***
cmdhEcDefParamValues	***cmpv***
cmdhLimits	***cml***
cmdhNetworkAccessRules	***cmnr***
cmdhNwAccessRule	***cmwr***
cmdhBuffer	***cmbf***

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 types members short names

Parameter 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***
resourceType	filterCriteria	***rty***
sizeAbove	filterCriteria, eventNotificationCriteria	***sza***
sizeBelow	filterCriteria, eventNotificationCriteriay	***szb***
contentType	filterCriteria	***cty***
limit	filterCriteria	***lim***
attribute	filterCriteria, eventNotificationCriteria	***atr***
resourceStatus	eventNotificationCriteria, notificationEvent	***rss***
operationMonitor	eventNotificationCriteria, notificationEvent	***om***
filterUsage	filterCriteria	***fu***
eventCatType	eventCat	***ect***
eventCatNo	eventCat	***ecn***
number	batchNotify	***num***
duration	batchNotify	***dur***
notification	aggregatedNotification	***sgn***
notificationEvent	notification	***nev***
verificationRequest	notification	***vrq***
subscriptionDeletion	notification	***sud***
subscriptionReference	notification	***sur***
creator	notification	***cr****
notificationForwardingURI	notification	***nfu****
operation	operationMonitor	***opr***
originator	operationMonitor	***org***
accessId	externalID	***aci***
MSISDN	externalID	***msd***
action	actionStatus	***acn***
status	actionStatus	***sus***
childResource	All except execInstance, announced resource, management resources from firmware	***ch***
aggregatedNotification	Primitive Content	***agn***
aggregatedResponse	Primitive Content	***agr***

NOTE: * marked short names have been already assigned in attribute Table 8.2.3-1.

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 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-v1_0_0.xsd and CDT-responsePrimitive-v1_0_0.xsd, respectively. The data types used in these XSD files comply with the definitions in clause 6 and clause 7 of this specification.

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 this specification).

NOTE: XML Schema files that use short names might be made available at a future date.

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 7.4.1.1 and the datatype definition given in CDT-notification-v1_0_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:req 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-v1_0_0.xsd">
   <op>1</op>
   <to>//cse1.mym2msp.org/</to>
   <fr>//cse1234/app567</fr>
   <ri>0002bf63</ri>
   <ty>4</ty>
   <pc>
     <cin rn="temp754">
       <cnf>application/xml:1</cnf>
       <con>PHRpbWU+MTc4ODkzMDk8L3RpbWU+PHRlbXA+MjA8L3RlbXA+DQo=</con>
     </cin>
   </pc>
</m2m:req>
```

The XML elements have the following meaning:

- req: 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 m2m:anyURI: URI of the target resource.
- fr: ***From*** parameter of type m2m:ID: ID of the Originator (either AE-ID or CSE-ID).
- ri: ***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. The instance name is given in the XML name attribute, here rn="temp754". In this example, the cn parameter includes an instance of a <contentInstance> resource which consists of two attributes: contentInfo (cnf) – which specifies base64 encoding - and the content (con) itself.

8.4 JSON serialization

8.4.1 Terminology

The following conventions are used in the clause that follows.

- The italicized terms *object*, *member*, *name*, *array*, *number*, *string*, *boolean* and *null* are to be interpreted as in RFC 7159 [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 RFC 7159 [19]. This JSON *object* shall be restricted to Unicode characters defined in The Unicode Standard and encoded using UTF-8 as described in 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 this specification, as follows:

1. All *member*'s *names* shall be the short name defined in clause 8.1.
2. If an *element* is defined in this specification 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 8.1.
3. The membership of each nested *object* shall respect the cardinality constraints from the corresponding XSD complex type definition,
4. If an *element* is defined in this specification as having an atomic data type that is numeric in nature (e.g. xs:integer or a type derived from it) then its value is serialized into the JSON *member* as a *number*.
5. 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*.
6. 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*.
7. 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*.
8. 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.
9. If an *element* is defined as having maxOccurs > 1 in the corresponding XSD then its parent JSON *member* is serialized as an *array*.
10. 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 *name* "val".
11. The *members* (at each level) may be serialized in any order. The order in which they appear in the corresponding XSD file is immaterial.

The ***Content*** parameter is treated just like any other parameter of complex type. It is serialized as an object and its

members are the attributes and/or child resource references of the Resource that is being transferred. The ***Content*** parameter is not required to contain all the attributes of the Resource.

8.4.3 Examples

Here is an example that shows the payload of a request message serialized using JSON:

{“op”: “1”, “fr”: “//xxxxx/2345”, “to”: “//xxxxx/99”, “ri”: “A1234”, “pc”: {“se”: “* 0-5 2,6,10 * * *”}, “ty”: 20}

- op: operation (in this case it’s Create)
- fr: ID of the Originator (either the AE or CSE)
- to: URI of the target resource
- ri: request identifier (this is a string)
- pc: attributes of the resource to be provided by 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.

Annex A(void):

Annex B(normative): Device triggering

B.1. Providing device triggering service by means of 3GPP networks

B.1.1. Introduction

3GPP Underlying Network has defined a dedicated interface for requesting device triggering. The normative references for applicable interfaces are as follows: 3GPP TS 23.682 [15]. The specification for the interface Tsp is described in 3GPP TS 29.368 [16]. Tsp interface uses Diameter Base Protocol as specified in IETF RFC 3588 [13], in order to use such an interface the CSE shall act as a Diameter client as described in IETF RFC 6733 [14].

Editor's Note: IETF RFC 3588 Reference needs to be checked to determine that it is current.

Before the CSE initiates the device triggering, the CSE and MTC-IWF shall execute the procedures once as specified in 3GPP TS29.368 [16].

B.1.2. Device action request command

When a CSE needs to issue a device triggering request to the MTC-IWF, the CSE shall send a Device-Action-Request (DAR) command (for detail, see TS 29.368 [16]). The following list provides the parameters mapping between the oneM2M and 3GPP.

Either External-Id or MSISDN: the CSE maps it to the M2M-Ext-ID, see clause 6.2.

SCS identifier: the CSE maps it to the CSE-ID, see clause 6.2.

Application Port Identifier: the CSE maps it to Trigger-Recipient-ID, see clause 6.2.

B.1.3. Device action answer command

As a result of device triggering request to MTC-IWF, the CSE receives a Device-Action-Answer (DAA) command (for detail, see TS 29.368 [16]).

B.1.4. Device notification request command

As a report of the result for device triggering delivery by 3GPP network, the CSE receives a Device-Notification-Request (DNR) command (for detail, see TS 29.368 [16]).

B.1.5. Device notification answer command

As a result of device notification request to MTC-IWF, the CSE sends a Device-Notification-Answer (DNA) command (for detail, see TS 29.368 [16]).

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, 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 xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.onem2m.org/xml/protocols"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   elementFormDefault="unqualified"
   <xs:include schemaLocation="CDT-commonTypes-v1_0_0.xsd" />
   <xs:include schemaLocation="CDT-contentInstance-v1_0_0.xsd" />
   <xs:include schemaLocation="CDT-subscription-v1_0_0.xsd" />

   <xs:element name="container">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from announceableResource -->
            <xs:extension base="m2m:announceableResource">
               <!-- Resource Specific Attributes -->
               <xs:sequence>
                  <xs:element name="stateTag" type="xs:nonNegativeInteger" />
                  <xs:element name="creator" type="m2m:ID" />
                  <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" />

                <!-- Child Resources -->
                <xs:element name="latest" type="xs:anyURI" minOccurs="0" />
                <xs:element name="oldest" type="xs:anyURI" minOccurs="0" />
                <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:container" />
                    <xs:element ref="m2m:contentInstance" />
                    <xs:element ref="m2m:subscription" />
                  </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 Annex. 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-v1_0_0.xsd"
    name="12xx">
  <resourceType>3</resourceType>
  <resourceID>//IN-CSEID.m2m.myoperator.org/96719</resourceID>
  <parentID>//IN-CSEID.m2m.myoperator.org/96734</parentID>
  <creationTime>2013-12-31T12:00:00</creationTime>
  <lastModifiedTime>2013-12-31T12:00:00</lastModifiedTime>
  <labels>label1 label2</labels>
  <accessControlPolicyIDs >
    <accessControlPolicyID>l//IN-CSEID.m2m.myoperator.org/93405</accessControlPolicyID>
  </accessControlPolicyID/>
  <expirationTime>2013-12-31T12:30</labels>
  <stateTag>0 </labels>
  <creator>//IN-CSEID.m2m.myoperator.org/9125</creator>
  <maxNrOfInstances>5</maxNrOfInstances>
  <maxByteSize>104857600</maxByteSize>
  <maxInstanceAge>3600</maxInstanceAge>
  <currentNrOfInstances>2</currentNrOfInstances>
  <currentByteSize>6</currentByteSize>
  <latest>//IN-CSEID.m2m.myoperator.org/96739</latest>
  <locationID>//IN-CSEID.m2m.myoperator.org/1112</locationID>
  <ontologyRef>http://tempuri.org/ontologies/xyz</ontologyRef>
  <latest>//IN-CSEID.m2m.myoperator.org/96739</latest>
  <oldest>//IN-CSEID.m2m.myoperator.org/34722</oldest>

  <childResource name="instance1234" type="4">//IN-
CSEID.m2m.myoperator.org/1722</childResource>
  <childResource name="instance1235" type="4">//IN-
CSEID.m2m.myoperator.org/34722</childResource>
  <childResource name="1923" type="23">//IN-CSEID.m2m.myoperator.org/2323</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 sub-clauses of this Annex shall be created on the IN-CSE when the management request is performed using external management 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 Functional Architecture [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 Functional Architecture [6].

Table D.2-1: Data Type Definition of [firmware]

Data Type ID	File Name	Note
firmware, firmwareAnnc	CDT-firmware-v1_0_0.xsd	

Table D.2-2: Resource specific attributes of [firmware]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1001 (firmware)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
version	M	O	xs:string	
name	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

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.2.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.1.2.2.

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 7.1.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 “Create/Update/Retrieve/Delete/Notify operation is performed”:

When the attribute *update* of the firmware resource is updated to TRUE, use the downloaded firmware image to update the current using firmware. The Receiver may need to update the *fwVersion* attribute of the [deviceInfo] resource if needed.

D.2.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.2.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.1.2.2:

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 TS-0001 Functional Architecture [6].

Table D.3-1: Data Type Definition of [software]

Data Type ID	File Name	Note
software, softwareAnnc	CDT-software-v1_0_0.xsd	

Table D.3-2: Resource specific attributes of [software]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1002 (software)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
version	M	O	xs:string	
name	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

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> resource specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.3.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

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 7.1.2.1.

Receiver:

Primitive specific operation additional to Recv-6.5 “Create/Update/Retrieve/Delete/Notify operation is performed”:

When the attribute *install* of the [software] resource is updated to TRUE, install the 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 *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.

The Receiver may need to update the s*wVersion* attribute of the [deviceInfo] resource if needed.

D.3.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.3.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

Primitive specific step after generic procedure defined in clause 7.1.2.2.

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 TS-0001 Functional Architecture [6].

Table D.4-1: Data Type Definition of [memory]

Data Type ID	File Name	Note
memory, memoryAnnc	CDT-memory-v1_0_0.xsd	

Table D.4-2: Resource specific attributes of [memory]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1003 (memory)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
memAvailable	M	O	xs:unsignedLong	Unit: Byte.
memTotal	M	O	xs:unsignedLong	Unit: Byte.

D.4.2. Resource specific procedure on CRUD operations

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.4.2.1. Create

.***Originator:***

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.4.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.4.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.4.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.5. Resource [areaNwkInfo]

D.5.1. Introduction

The detailed description of the [areaNwkInfo] resource can be found in clause D.5 of TS-0001 Functional Architecture [6].

Table D.5-1: Data Type Definition of [areaNwkInfo]

Data Type ID	File Name	Note
areaNwkInfo, areaNwkInfoAnnc	CDT-areaNwkInfo-v1_0_0.xsd	

Table D.5-2: Resource specific attributes of [areaNwkInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1004 (areaNwkInfo)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
areaNwkType	M	O	xs:string	
listOfDevices	M	O	list of xs:anyURI	

D.5.2. Resource specific procedure on CRUD operations

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.5.2.1. Create

.***Originator:***

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.5.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.5.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.5.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.6. Resource [areaNwkDeviceInfo]

D.6.1. Introduction

The detailed description of the [areaNwkDeviceInfo] resource can be found in clause D.6 of TS-0001 Functional Architecture [6].

Table D.6-1: Data Type Definition of [areaNwkDeviceInfo]

Data Type ID	File Name	Note
areaNwkDeviceInfo, areaNwkDeviceInfoAnnc	CDT-areaNwkDeviceInfo-v1_0_0.xsd	

Table D.6-2: Resource specific attributes of [areaNwkDeviceInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1005 (areaNwkDeviceInfo)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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
status	O	O	xs:string	
listOfNeighbors	M	O	list of xs:anyURI	

D.6.2. Resource specific procedure on CRUD operations

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.6.2.1. Create

.***Originator:***

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.6.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

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 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.6.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.7. Resource [battery]

D.7.1. Introduction

The detailed description of the [battery] resource can be found in clause D.7 of Architecture TS-0001 [6].

Table D.7-1: Data Type Definition of [battery]

Data Type ID	File Name	Note
battery, batteryAnnc	CDT-battery-v1_0_0.xsd	

Table D.7-2: Resource specific attributes of [battery]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1006 (battery)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
batteryLevel	M	O	xs:unsignedInt	Range: 0-100 Unit: percent
batteryStatus	M	O	m2m:batteryStatus	

D.7.2. Resource specific procedure on CRUD operations

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.7.2.1. Create

.***Originator:***

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.7.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.7.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.7.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

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 Architecture TS-0001 [6].

Table D.8-1: Data Type Definition of [deviceInfo]

Data Type ID	File Name	Note
deviceInfo, deviceInfoAnnc	CDT-deviceInfo-v1_0_0.xsd	

Table D.8-2: Resource specific attributes of [deviceInfo]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1007 (deviceInfo)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
deviceLabel	M	O	xs:string	
manufacturer	M	O	xs:string	
model	M	O	xs:string	
deviceType	M	O	xs:string	
fwVersion	M	O	xs:string	
swVersion	M	O	xs:string	
hwVersion	M	O	xs:string	

D.8.2. Resource specific procedure on CRUD operations

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.8.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.8.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.8.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.8.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

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 TS-0001 Functional Architecture [6].

Table D.9-1: Data Type Definition of [deviceCapability]

Data Type ID	File Name	Note
deviceCapability, deviceCapabilityAnnc	CDT-deviceCapability-v1_0_0.xsd	

Table D.9-2: Resource specific attributes of [deviceCapability]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1008 (deviceCapability)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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 the Table 6.3.2.31
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

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.9.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.9.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

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.

D.9.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.9.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

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 TS-0001 Functional Architecture [6].

Table D.10-1: Data Type Definition of [reboot]

Data Type ID	File Name	Note
reboot, rebootAnnc	CDT-reboot-v1_0_0.xsd	

Table D.10-2: Resource specific attributes of [reboot]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1009 (reboot)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.10.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.10.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

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..

D.10.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.10.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

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 TS-0001 Functional Architecture [6].

Table D.11-1: Data Type Definition of [eventLog]

Data Type ID	File Name	Note
eventLog, eventLogAnnc	CDT-eventLog-v1_0_0.xsd	

Table D.11-2: Resource specific attributes of [eventLog]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1010 (eventLog)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
logTypeId	M	O	m2m:logTypeId	See Table 6.3.3.2.23-1
logData	M	O	xs:string	the content and format of this attribute is out of this specification.
logStatus	M	O	m2m:logStatus	See Table 6.3.3.2.24-1
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

When management is performed using external management technologies, the procedures defined in 7.3.15.2 <mgmtObj> specific procedures shall be used. The following clauses define additional procedures besides the generic procedure defined in 7.1.2.

D.11.2.1. Create

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.11.2.2. Update

Originator:

No change from the generic procedures in clause 7.1.2.1.

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.

D.11.2.3. Retrieve

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.11.2.4. Delete

Originator:

No change from the generic procedures in clause 7.1.2.1.

Receiver:

No change from the generic procedures in clause 7.1.2.2.

D.12. Resource [cmdhPolicy]

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 TS-0001 Functional Architecture [6].

Table D.12-1: Data Type Definition of [cmdhPolicy]

Data Type ID	File Name	Note
cmdhPolicy	CDT-cmdhPolicy-v1_0_0.xsd	

Note that the optional <subscription> child resources are not used for CMDH policies.

Table D.12-2: Resource specific attributes of [cmdhPolicy]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1011 (cmdhPolicy)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
name	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 7.3.15 for the generic <mgmtObj> resource type apply.

D.12.1. 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 TS-0001 Functional Architecture [6].

Table D.12.1-1: Data Type Definition of [activeCmdhPolicy]

Data Type ID	File Name	Note
activeCmdPolicy	CDT-activeCmdhPolicy-v1_0_0.xsd	

Table D.12.1-2: Resource specific attributes of [activeCmdhPolicy]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1012 (activeCmdhPolicy)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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.12.2. 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 TS-0001 Functional Architecture [6].

Table D.12.2-1: Data Type Definition of [cmdhDefaults]

Data Type ID	File Name	Note
cmdhDefaults	CDT-cmdhDefaults-v1_0_0.xsd	

Table D.12.2-2: Resource specific attributes of [cmdhDefaults]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1013 (cmdhDefaults)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
mgmtLink	M	O	m2m:mgmtLinkRef	1 or more link(s) to [cmdhDefEcValue] resource instance(s)

D.12.3. 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 TS-0001 Functional Architecture [6].

Table D.12.3-1: Data Type Definition of [cmdhDefEcValue]

Data Type ID	File Name	Note
cmdhDefEcValue	CDT-cmdhDefEcValue-v1_0_0.xsd	

Table D.12.3-2: Resource specific attributes of [cmdhDefEcValue]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1014 (cmdhDefEcValue)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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.12.4. 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 TS-0001 Functional Architecture [6].

Table D.12.4-1: Data Type Definition of [cmdhEcDefParamValues]

Data Type ID	File Name	Note
cmdhEcDefParamValues	CDT-cmdhEcDefParamValues-v1_0_0.xsd	

Table D.12.4-2: Resource specific attributes of [cmdhEcDefParamValues]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1015 (cmdhEcDefParamValues)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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.12.5. Resource [cmdhLimits]

The resource [cmdhLimits] represents limits for CMDH related parameter values. The detailed description can be found in clause D.12.5 of TS-0001 Functional Architecture [6].

Table D.12.5-1: Data Type Definition of [cmdhLimits]

Data Type ID	File Name	Note
cmdhLimits	CDT-cmdhLimits-v1_0_0.xsd	

Table D.12.5-2: Resource specific attributes of [cmdhLimits]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1016 (cmdhLimits)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
order	M	O	xs:positiveInteger	None
requestOrigin	M	O	m2m:listOfM2MID	None
requestContext	O	O	xs:string	None
requestContextNotification	O	O	xs:boolean	None
requestCharacteristics	O	O	xs:string	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	m2m: permittedBooleanValues	None

D.12.6. 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 TS-0001 Functional Architecture [6].

Table D.12.6-1: Type Definition of [cmdhNetworkAccessRules]

Data Type ID	File Name	Note
cmdhNetworkAccessRules	CDT-cmdhNetworkAccessRules-v1_0_0.xsd	

Table D.12.6-2: Resource specific attributes of [cmdhNetworkAccessRules]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1017 (cmdhNetworkAccessRules)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
applicableEventCategories	M	O	list of m2m:eventCatWithDef	Exactly one instance of this [cmdhNetworkAccessRules] 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.12.7. 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 TS-0001 Functional Architecture [6].

Table D.12.7-1: Data Type Definition of [cmdhNwAccessRule]

Data Type ID	File Name	Note
cmdhNwAccessRule	CDT-cmdhNwAccessRule-v1_0_0.xsd	

Table D.12.7-2: Resource specific attributes of [cmdhNwAccessRule]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1018 (cmdhNwAccessRule)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
targetNetwork	M	O	m2m:listOfM2MID	None
minReqVolume	M	O	xs:nonNegativeInteger	Unit: byte
backOffParameters	M	O	m2m: backOffParameters	Ordered sequence of 3 values: backoffTime, backoffTimeIncrement, maximumBackoffTime, Unit: ms
otherConditions	O	O	xs:anyType	None
mgmtLink	M	O	m2m:mgmtLinkRef	Link to an instance “allowedSchedule” of a <schedule> resource

D.12.8. 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 TS-0001 Functional Architecture [6].

Table D.12.8-1: Data Type Definition of [cmdhBuffer]

Data Type ID	File Name	Note
cmdhBuffer	CDT-cmdhBuffer-v1_0_0.xsd	

Table D.12.8-2: Resource specific attributes of [cmdhBuffer]

Attribute Name	Request Optionality		Data Type	Default Value and Constraints
	Create	Update		
mgmtDefinition	M	NP	See clause 7.3.15	1019 (cmdhBuffer)
objectID	O	NP	See clause 7.3.15	
objectPaths	O	NP	See clause 7.3.15	
description	O	O	See clause 7.3.15	
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:

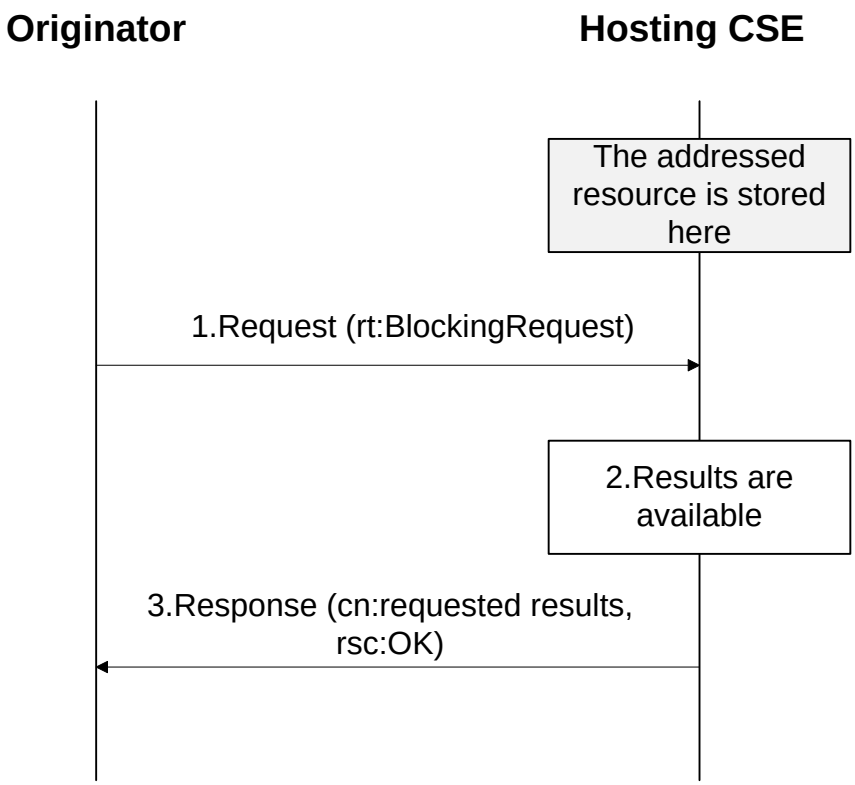

Figure E.1-1: Blocking access to resource

1. The Originator sends a request to accessing 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 annex 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.

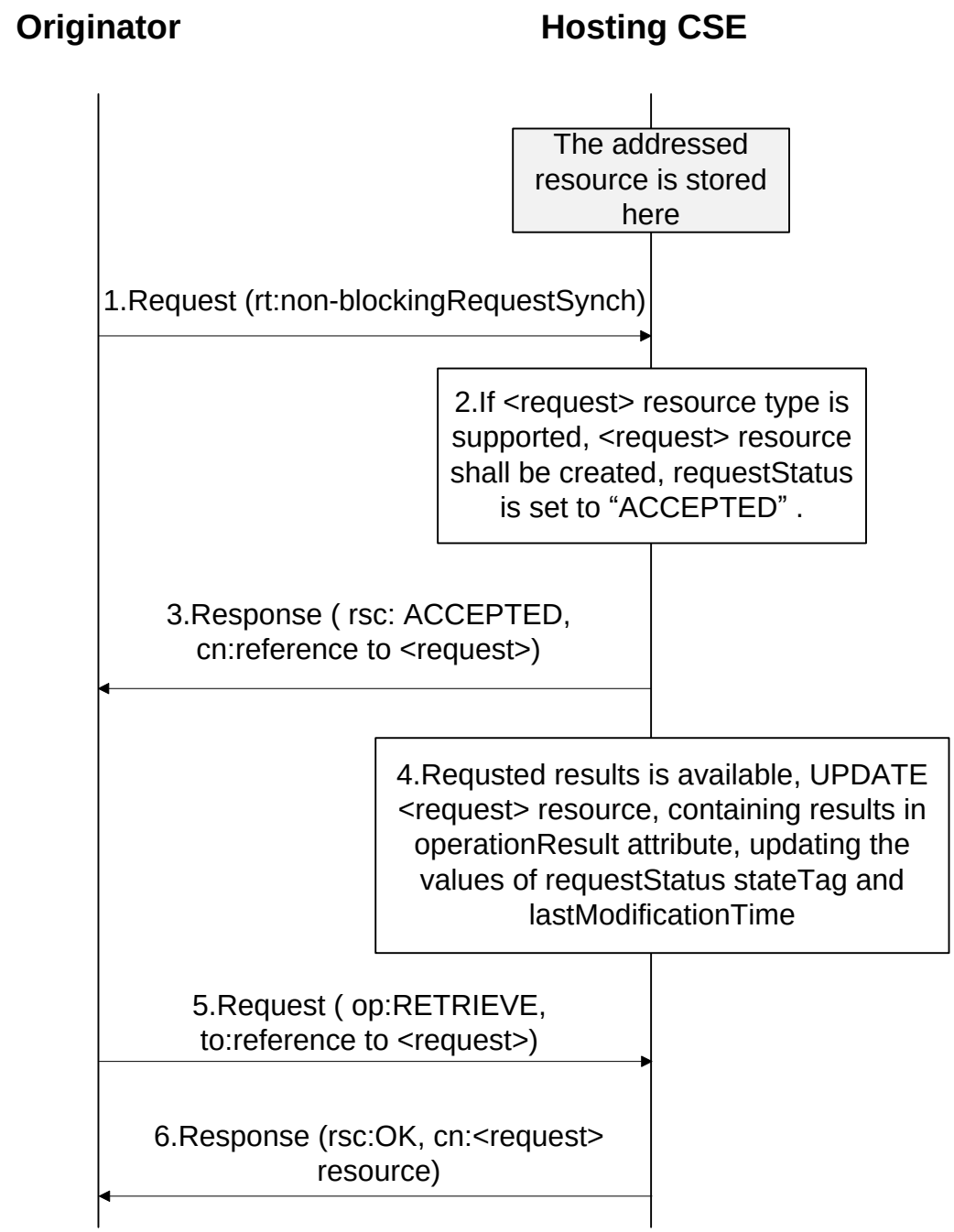

Figure E.2.2-1: non-Blocking access to resource in synchronous mode (no hop)

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 <request> resource. The *requestStatus* attribute of the <request> resource is set to “ ACCEPTED”. Please refer to Table 7.1.2.2.4-1 and Table 7.1.2.2.4-2for 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. :

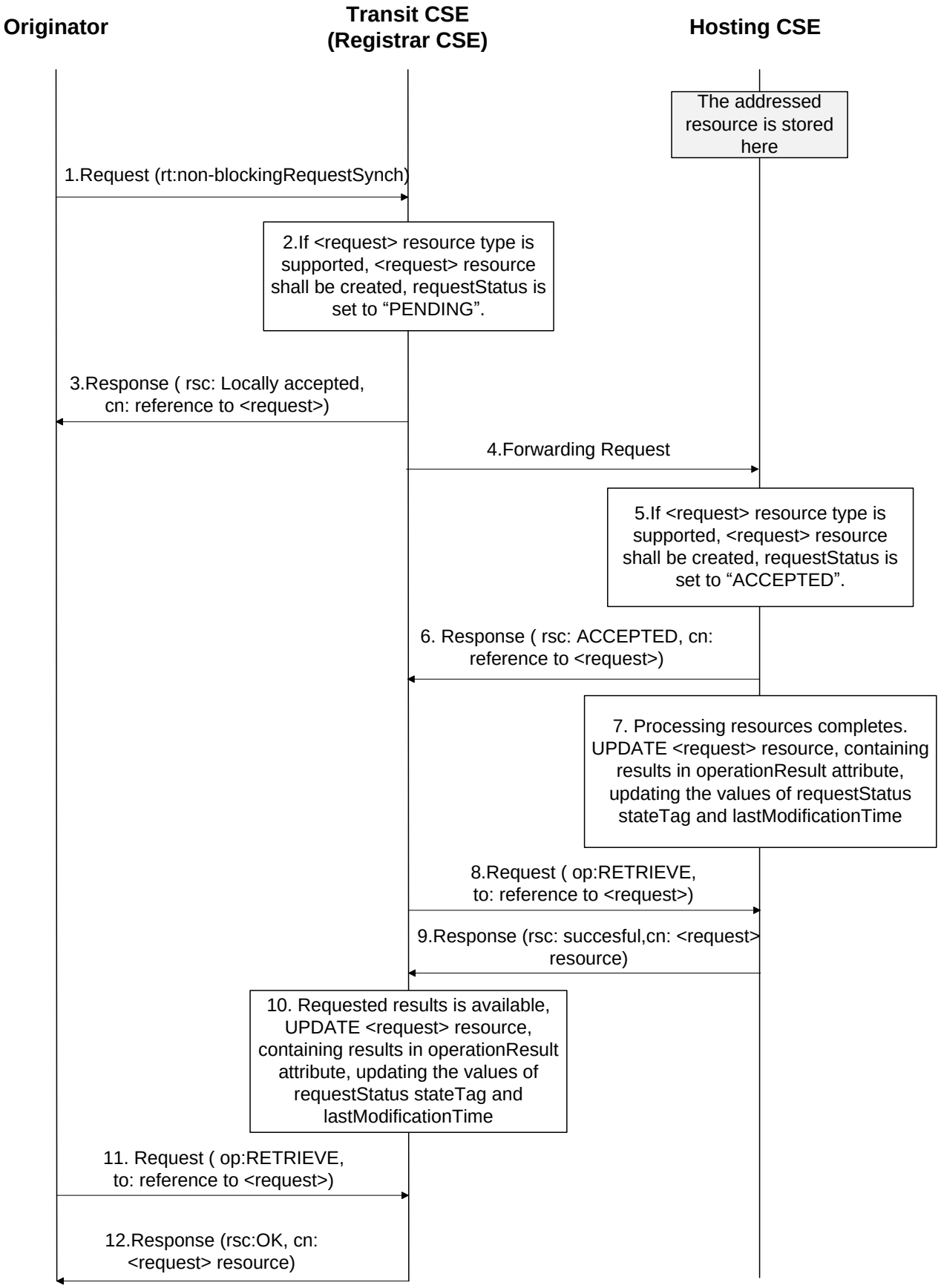

Figure E.2.2-2: non-Blocking access to resource in synchronous mode (one hop)

1. The Originator sends a request to its Registrar CSE (this is aTransit CSE, not the Hosting CSE), setting the Response Type parameter of the request to ‘nonblockingRequestSynch’.2. If the Transit CSE supports non-blocking synchronouse 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 “ACCEPTED”. The Please refer to Table 7.3.12.1-3 for other attributes.

2. 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.

3. The Transit CSE forwards the original request to the Hosting CSE.

4. If the Hosting CSE supports non-blocking synchronouse 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 “ACCEPTED”. Please refer to Table 7.1.2.2.4-1 and Table 7.1.2.2.4-2for other attributes..

5. 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***.

6. The Hosting CSE processes the resource according to the requested operation. When the requested operation has finihsed, 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*.

7. The Transit CSE requests to RETRIEVE the original requested results by addressing the <request> resource

8. The Hosting CSE sends a response to the Transit CSE. The response contains the <request> resource as its ***Content***.

9. 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.*

10. The Originator requests to RETRIEVE the original requested results by addressing the <request> resource.

11. 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 one M2M 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 minimising 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 TS-0001 Functional Architecture [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 "name" 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 ***name*** 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 6.3.1 or 6.3.2.

   b. A list of one of the data types listed in clause 6.3.1 or 6.3.2. If the list type is not already included in 6.3.2 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 6.3.1 or 6.3.2. This may be added to clause 6.3.2, 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 6.3.1 or 6.3.2.

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 6.3.2, 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-v1_0_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 7.3.

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, it shall be avoided to use identical names. It is recommended to use the text suffix 'Type' in data type names. Example: `<xs:element name="status" type="m2m:statusType />`

12) In cases where a Resource has an optional read/write attribute, that attribute should be marked as xsi:nillable in the schema. This is to allow a requester to delete the attribute by supplying a nil value for it. If the resource is subsequently retrieved, the deleted attribute will no longer be included in the resource.

13) 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

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 7.3.10.

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.1. Location request by means of OMA-REST-NetAPI-TerminalLocation interface

G.1.1. 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 the clause G.1.3. .

G.1.2. 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 G.1.3.).

The OMA REST NetAPI for Terminal Location allows CSE to obtain information about geographical location of a terminal (e.g. Node in oneM2M architecture TS-0001 Functional Architecture [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 the annex E of TS-0003) 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].

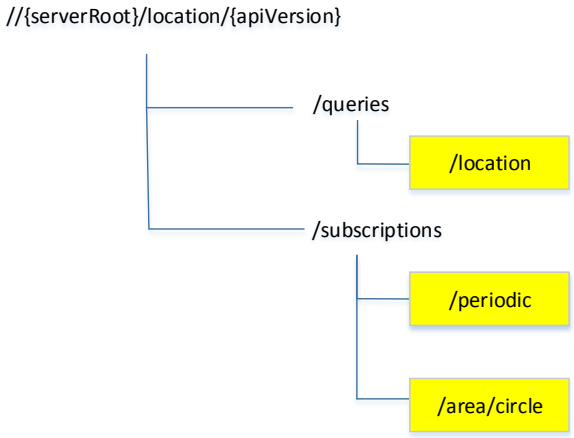

Figure G.1.2-1: Resource Structure defined by NetAPI for Terminal Location

The two capabilities used for oneM2M system location request are 'Terminal location'. 'Periodic location notification subscriptions' and 'area notification subscriptions'. The table below describes the URL structure, data structure and mapping with CRUD operation of each resource.

Table G.1.2-3: 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 the table above, 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. The table also contains the relevant attributes column that is correlated with either <locationPolicy> or <accessControlPolicy> resource type defined (3GPP TS 23.003 [17]). Only attributes that may be utilized by oneM2M system are described. For the detailed information, see the [28].

Table G.1.2-4: 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.1.2-5: 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.	*locationUpdatePeriod* in the <locationPolicy> resource type

Table G.1.2-6: 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.	

G.1.3. Procedures for terminal location

G.1.3.1. Request in a single query toward a location server

This procedure shows how to request and return location for a M2M Node.

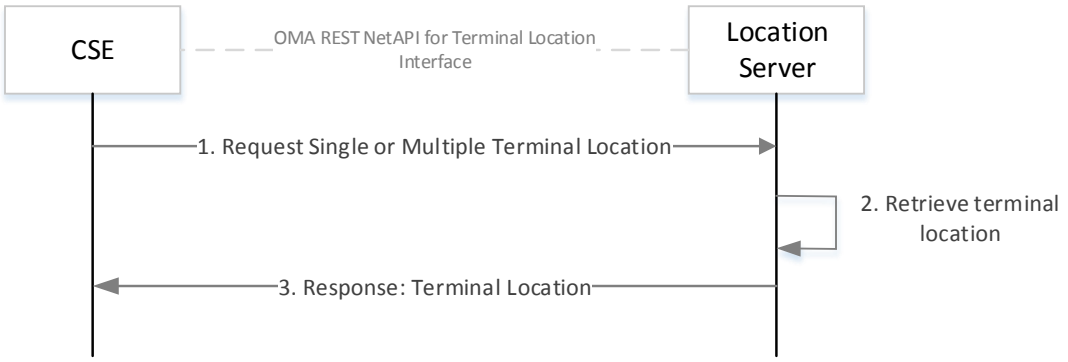

Figure G.1.3.1-1: Single Query Toward Location Server

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.1.2-3 shall be used with RETRIEVE operation.

   NOTE: GET operation shall be 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.1.4. Subscribe to notifications for periodic location updates

This procedure shows how to control subscriptions for periodic notifications about terminal location.

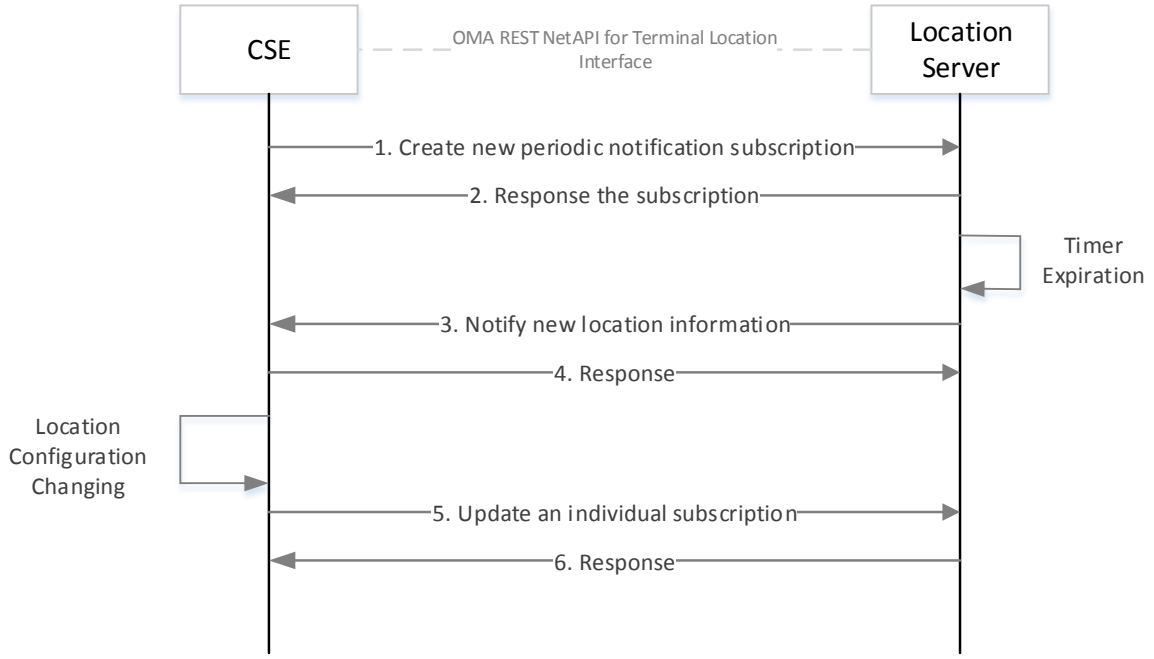

Figure G.1.4-1: Subscribe to Notification for Periodic Location Updates

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.1.2-3 shall be used with CREATE operation.

   NOTE: POST operation shall be 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: 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: POST operation shall be 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.1.2-3 shall be used with UPDATE operation.

NOTE: PUT operation shall be used for this UPDATE operation.

G.1.5. Subscribe to notifications for area updates

This procedure shows how to subscribe to area update notification.

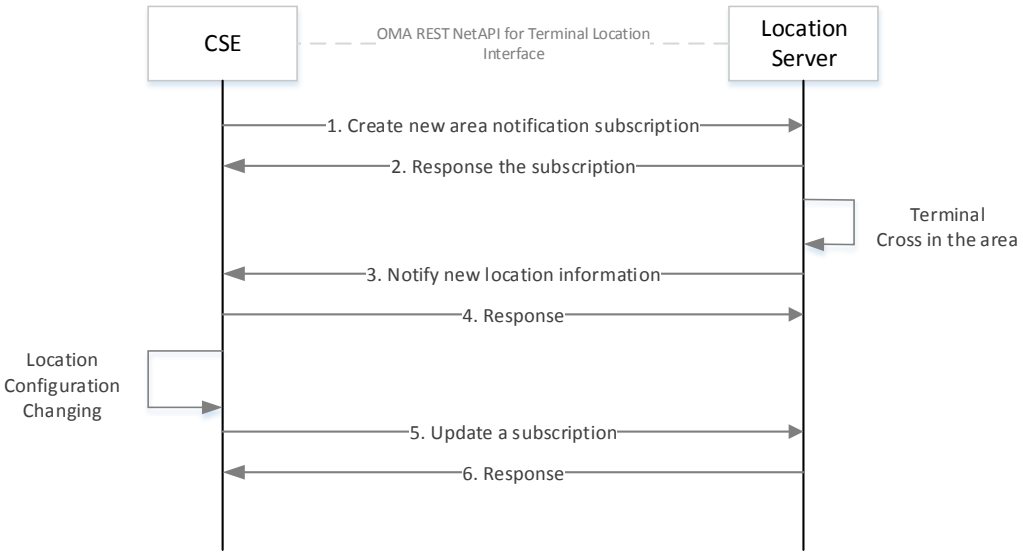

Figure G.1.5-1: Subscribe to Notification for Area Updates

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 the table-G.1-3 shall be used with CREATE operation.
   NOTE: POST operation shall be 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: Alternatively, the hosting CSE obtains the notifications using a Notification Channel [i.3].

   NOTE: POST operation shall be 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 the table-G.1-3 shall be used with UPDATE operation.

   NOTE: PUT operation shall be used for this UPDATE operation.

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:

- ***ec***: Event Category
- ***rqet***: Request expiration time
- ***rset***: Result expiration time
- ***oet***: operation execution time
- ***rp***: result persistence
- ***da***: delivery aggregation

CMDH-related response message parameters are:

- ***ec***: Event Category
  - ***'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.
- ***rset***: Result expiration time
- ***da***: delivery aggregation
  - 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 the next clauses.

In the following description it is assumed that the CSE behavior 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 [cdmhPolicy] 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 re-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 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 7.1.2. *Description of Generic Procedures* of this TS. 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 Annex 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' value 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 7.2.1.4.

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:

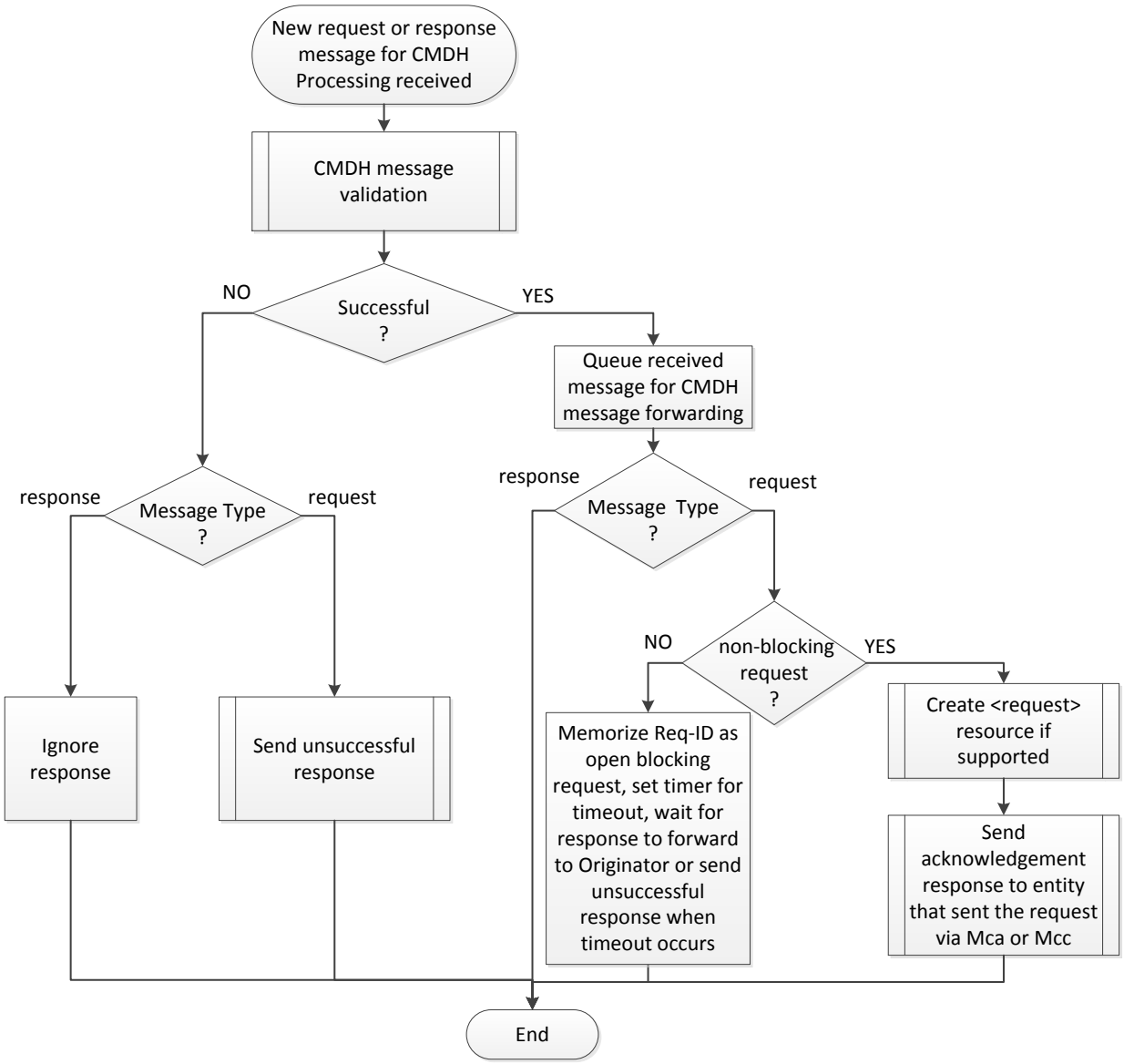

Figure H.2.2-1: CMDH Processing

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 default 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 with a duration, the receiving CSE shall translate the values of those parameters into absolute times 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 of '**bestEffort'**, use the following values for any missing CMDH-related parameters: For a request message use '***rqet***' = 'infinite', '***rset***' = 'infinite', '***oet***' = 'now', '***rp***' = 'none', '***da***' = ON. For a response message use '***rset***' = 'infinite', '***da***' = ON. Continue with step 2.

1.2.3. If the message parameter '***ec***' has a value of '**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*'.

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.*

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 '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 '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 '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 '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***' is '**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 string *'default' in* the list defined by the '*applicableEventCategory*'.

3.2.3. Lookup all [cmdhNwAccessRule] child resources of *the* selected [cmdhNetworkAccessRules] resource

3.2.4. Among the 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 string *'default' in* the list defined by the '*applicableEventCategory*'.

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.

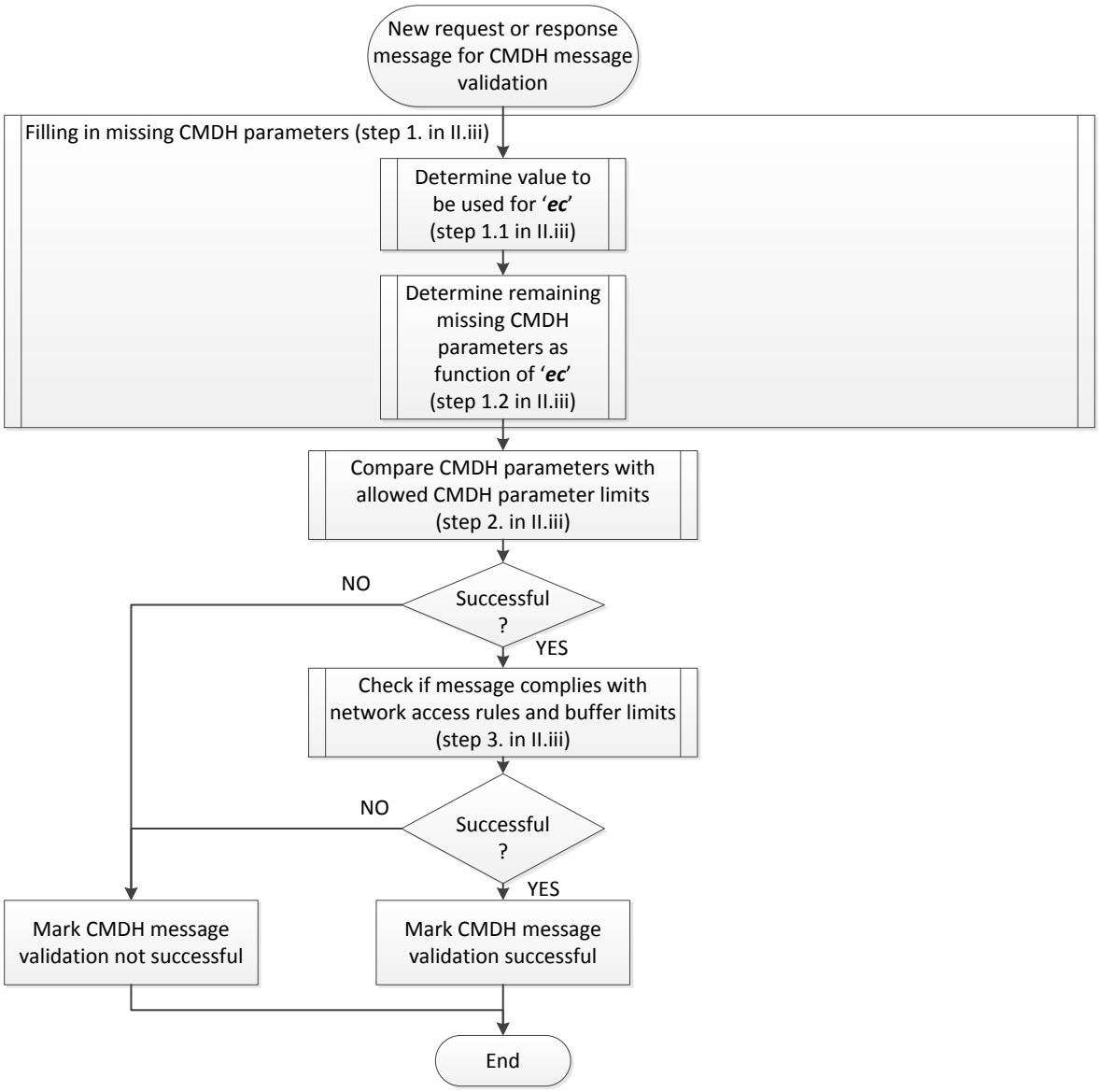

Figure H.2.3-1: CMDH message validation procedure

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 behavior. 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

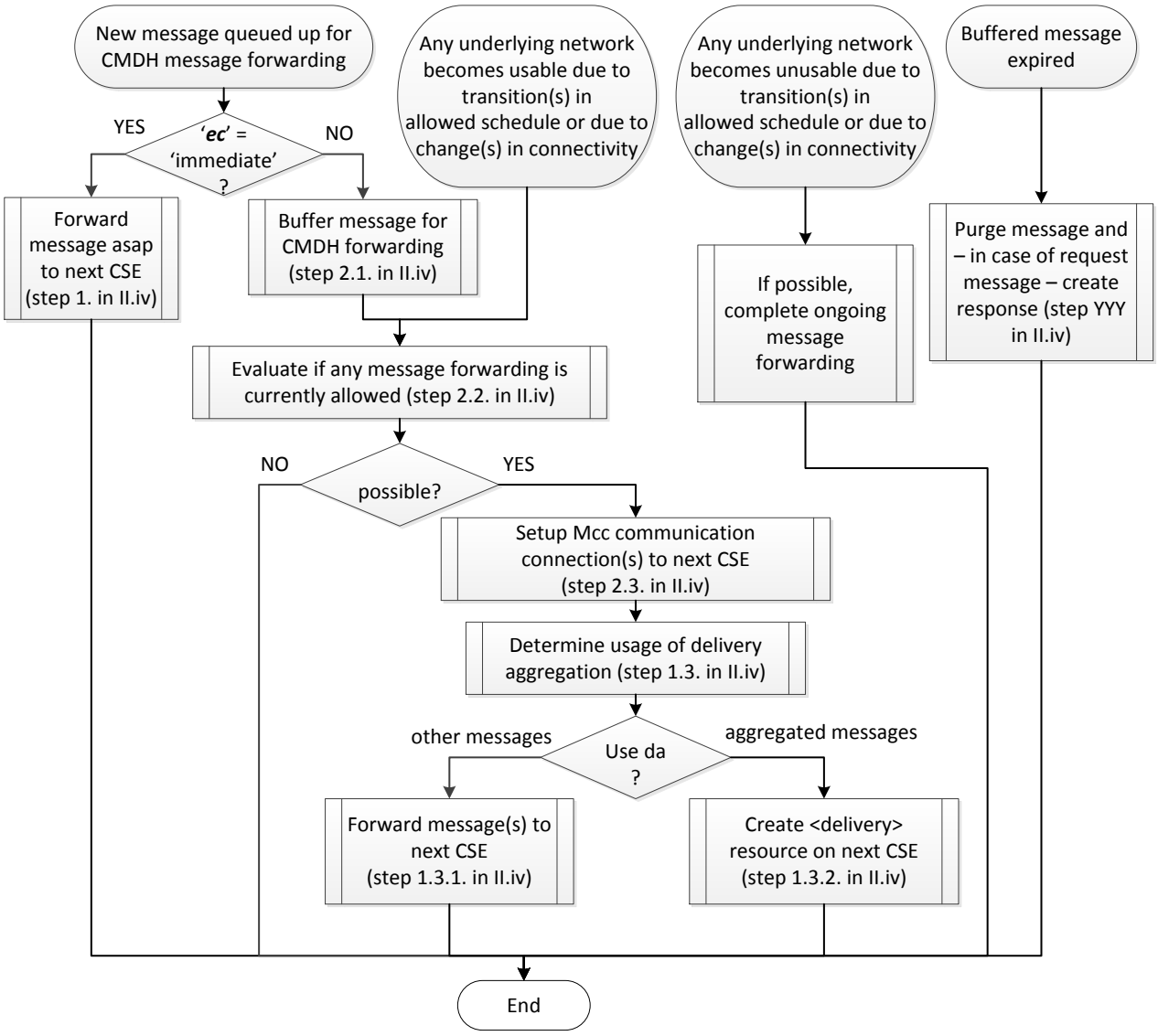

Figure H.2.4-1: CMDH message forwarding procedure

When a new message is getting queued up for CMDH message forwarding, carry out the following:

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 'ON', the Receiver CSE shall forward this message by creation of a <delivery> resource on the next CSE as outlined in clause 7.3.11. The receiver CSE can combine the forwarded message in the same <delivery> resource with other messages for which the '***da***' parameter set to 'ON' 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 '**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 below: If the message is a request message and the '***ec***' parameter of the messages has the value '**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*'.

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
- for which the corresponding target network could be used to reach the next CSE for forwarding the message under evaluation.

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 above for that message.

1.

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 above.

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.

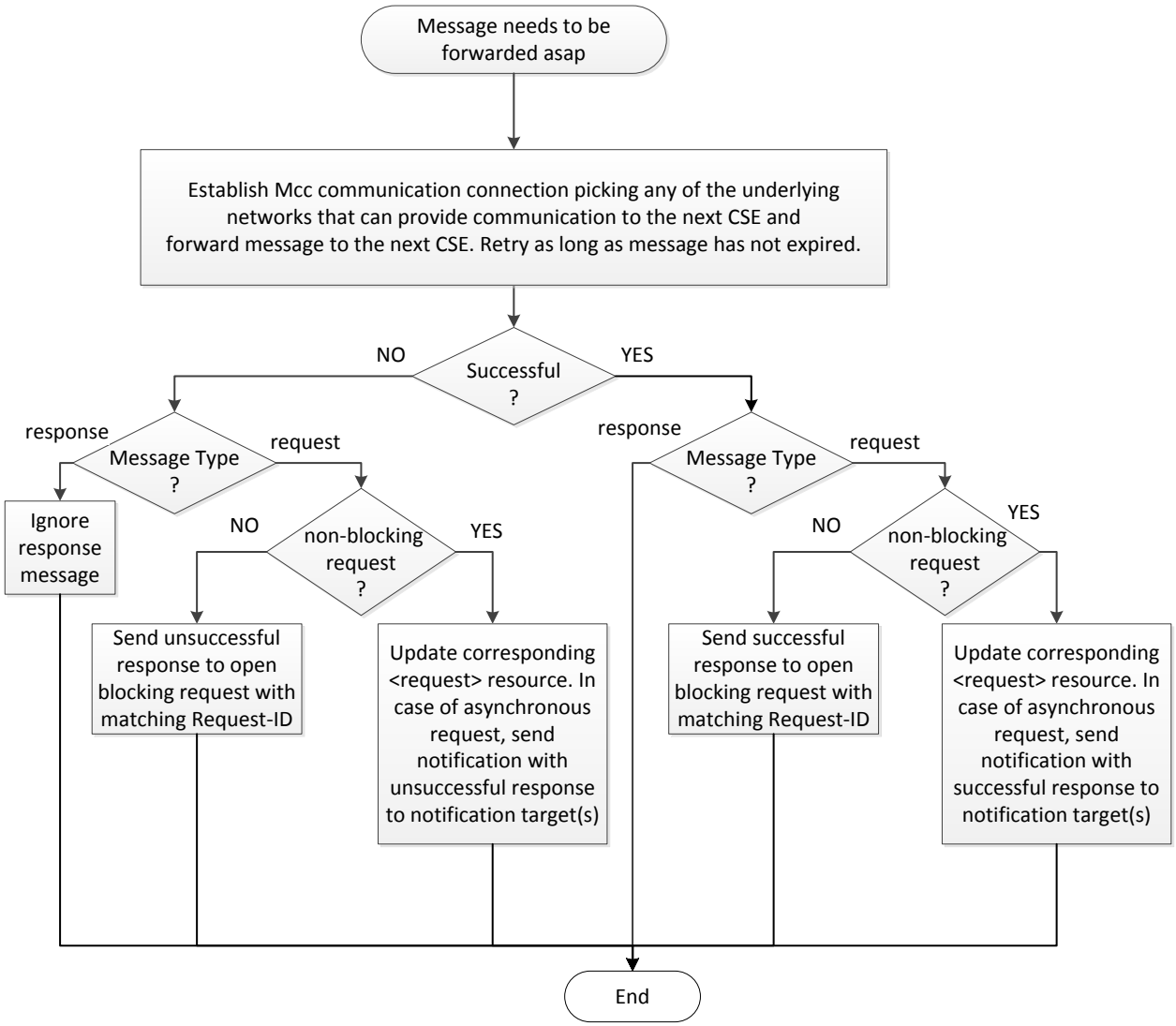

Figure H.2.4-2: Forwarding of messages with 'ec' = 'immediate'.

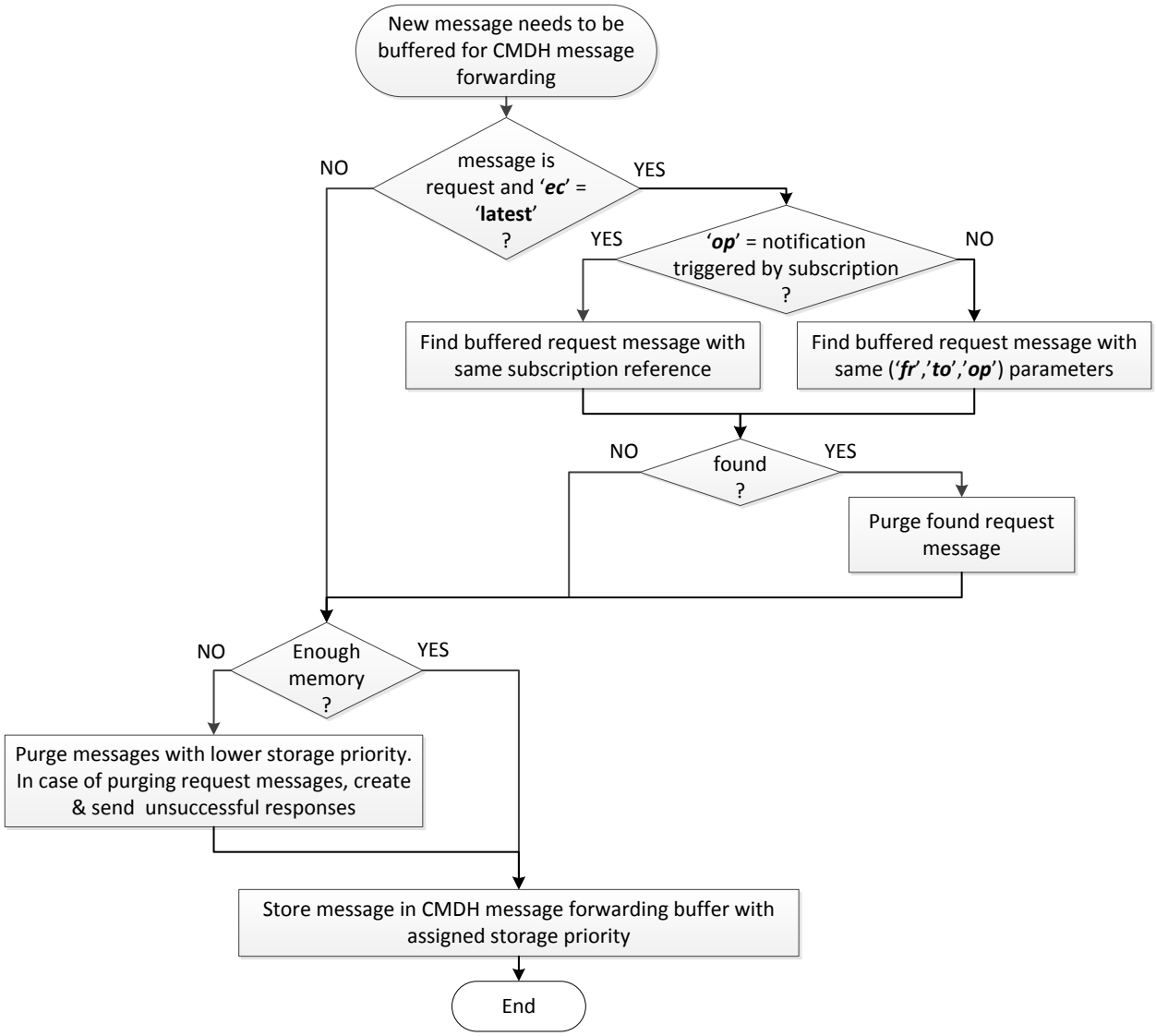

Figure H.2.4-3: Buffering of messages for CMDH message forwarding.

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 Security Solutions [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.

List of tables and figures

History

Publication history		
V1.0.1	30 Jan 2015	Release 1 - Publication