

**This draft contribution shows how TS-0033 could be updated to include Device Management IPE-based using SDT as introduced by the new release 5 work item WI-0109 “IPE-based Device Management with FlexContainers”**

Changes in existing sections

New section proposal §8: Device Management Operations

|  |
| --- |
| **oneM2M****Technical Specification** |
| Document Number | TS-0033-Vtbd |
| Document Name: | Interworking Framework |
| Date: | 2022 |
| Abstract: | This document is the specification describing interworking methodologies that are defined by oneM2M for the purpose of representing interactions with devices or functions in Proximal IoT networks that are not aware of oneM2M. This includes exposing non-oneM2M devices, applications and services to oneM2M entities via the oneM2M resource architecture, as well as exposing oneM2M functions and services to Proximal IoT networks that are not aware of oneM2M. This present document is independent of any specific Proximal IoT technology. Details for interworking with specific Proximal IoT Technologies are contained in other Technical Specifications. |
| Template Version: January 2017 (Do not modify) |

The present document is provided for future development work within oneM2M only. The Partners accept no liability for any use of this present document.

The present document has not been subject to any approval process by the oneM2M Partners Type 1. Published oneM2M specifications and reports for implementation should be obtained via the oneM2M Partners' Publications Offices.

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

Copyright Notification

© 2019, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC).

All rights reserved.

The copyright extends to reproduction in all media.

Notice of Disclaimer & Limitation of Liability

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

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

Contents

1 Scope 7

2 References 7

2.1 Normative references 7

2.2 Informative references 7

3 Definitions and abbreviations 8

3.1 Definitions 8

3.2 Abbreviations 8

5 Introduction 8

6 General interworking architecture 9

6.1 Concept of Representation 9

6.2 Role of IPE(s) 10

6.2.1 Fit with the Functional Architecture 10

6.2.2 Exposure of Proximal IoT functions to the oneM2M System 11

6.2.3 Exposure of native oneM2M functions to the Proximal IoT System 12

7 Representation of non-oneM2M entities in Proximal IoT networks 13

7.1 Representation of non-oneM2M Proximal IoT Devices 13

7.2 Representation of non-oneM2M Proximal IoT Applications 14

7.3 Representation of non-oneM2M Proximal IoT Services 15

8 Device Management Operations 16

8.1 [*flexNode*] management 16

8.1.0 Access Control 16

8.1.1 Create [*flexNode*] 16

8.1.1.1 Create <*subscription*> 17

8.1.1.2 Create DM SDT modules 17

8.1.2 Retrieve [*flexNode*] 17

8.1.3 Update [*flexNode*] 18

8.1.4 Delete [*flexNode*] 18

8.2 Generic DM SDT modules management 19

8.2.1 Create DM SDT *<flexContainer>* 19

8.2.1.1 Create <*subscription*> 19

8.2.2 Retrieve DM SDT *<flexContainer>* 20

8.2.3 Update DM SDT *<flexContainer >* 20

8.2.4 Delete DM SDT *<flexContainer >* 20

8.2.5 Notifications on *<flexContainer>* operations 21

8.3 Specific DM SDT modules management 21

8.3.1 Resource [*dmDeviceInfo*] 21

8.3.1.1 Introduction 21

8.3.1.2 Create 21

8.3.1.3 Retrieve 22

8.3.1.4 Update 22

8.3.1.5 Delete 22

8.3.1.6 Notification on update 22

8.3.2 Resource [*dmAgent*] 22

8.3.2.1 Introduction 22

8.3.2.2 Create 22

8.3.2.3 Retrieve 22

8.3.2.4 Update 22

8.3.2.5 Delete 22

8.3.2.6 Notification on update 23

8.3.2.7 Resource [*reboot*] 23

8.3.2.7.1 Introduction 23

8.3.2.7.2 Create 23

8.3.2.7.3 Retrieve 23

8.3.2.7.4 Update 23

8.3.2.7.5 Delete 23

8.3.2.7.6 Notification on update 23

8.3.2.8 Resource [*deployPackage*] 23

8.3.2.8.1 Introduction 23

8.3.2.8.2 Create 24

8.3.2.8.3 Retrieve 24

8.3.2.8.4 Update 24

8.3.2.8.5 Delete 24

8.3.2.8.6 Notification on update 24

8.3.3 Resource [*dmDataModelIO*] 24

8.3.3.1 Introduction 24

8.3.3.2 Create 24

8.3.3.3 Retrieve 25

8.3.3.4 Update 25

8.3.3.5 Delete 25

8.3.3.6 Notification on update 25

8.3.3.7 Resource [*readIO*] 25

8.3.3.7.1 Introduction 25

8.3.3.7.2 Create 25

8.3.3.7.3 Retrieve 25

8.3.3.7.4 Update 25

8.3.3.7.5 Delete 25

8.3.3.7.6 Notification on update 25

8.3.3.8 Resource [*writeIO*] 26

8.3.3.8.1 Introduction 26

8.3.3.8.2 Create 26

8.3.3.8.3 Retrieve 26

8.3.3.8.4 Update 26

8.3.3.8.5 Delete 26

8.3.3.8.6 Notification on update 26

8.3.4 Resource [*dmFirmware*] 26

8.3.4.1 Introduction 26

8.3.4.2 Create 27

8.3.4.3 Retrieve 27

8.3.4.4 Update 27

8.3.4.5 Delete 27

8.3.4.6 Notification on update 27

8.3.4.7 Resource [*updateFirmware*] 27

8.3.4.7.1 Introduction 27

8.3.4.7.2 Create 27

8.3.4.7.3 Retrieve 27

8.3.4.7.4 Update 28

8.3.4.7.5 Delete 28

8.3.4.7.6 Notification on update 28

8.3.4.8 Resource [*toggle*] 28

8.3.4.8.1 Introduction 28

8.3.4.8.2 Create 28

8.3.4.8.3 Retrieve 28

8.3.4.8.4 Update 28

8.3.4.8.5 Delete 28

8.3.4.8.6 Notification on update 28

8.3.5 Resource [*dmSoftware*] 29

8.3.5.1 Introduction 29

8.3.5.2 Create 29

8.3.5.3 Retrieve 29

8.3.5.4 Update 29

8.3.5.5 Delete 29

8.3.5.6 Notification on update 29

8.3.5.7 Resource [*activate*] 29

8.3.5.7.1 Introduction 29

8.3.5.7.2 Create 30

8.3.5.7.3 Retrieve 30

8.3.5.7.4 Update 30

8.3.5.7.5 Delete 30

8.3.5.7.6 Notification on update 30

8.3.5.8 Resource [*deactivate*] 30

8.3.5.8.1 Introduction 30

8.3.5.8.2 Create 30

8.3.5.8.3 Retrieve 30

8.3.5.8.4 Update 31

8.3.5.8.5 Delete 31

8.3.5.8.6 Notification on update 31

8.3.6 Resource [*dmPackage*] 31

8.3.6.1 Introduction 31

8.3.6.2 Create 31

8.3.6.3 Retrieve 31

8.3.6.4 Update 31

8.3.6.5 Delete 32

8.3.6.6 Notification on update 32

8.3.6.7 Resource [*install*] 32

8.3.6.7.1 Introduction 32

8.3.6.7.2 Create 32

8.3.6.7.3 Retrieve 32

8.3.6.7.4 Update 32

8.3.6.7.5 Delete 32

8.3.6.7.6 Notification on update 32

8.3.6.8 Resource [*uninstall*] 32

8.3.6.8.1 Introduction 32

8.3.6.8.2 Create 33

8.3.6.8.3 Retrieve 33

8.3.6.8.4 Update 33

8.3.6.8.5 Delete 33

8.3.6.8.6 Notification on update 33

8.3.6.9 Resource [*update*] 33

8.3.6.9.1 Introduction 33

8.3.6.9.2 Create 33

8.3.6.9.3 Retrieve 33

8.3.6.9.4 Update 34

8.3.6.9.5 Delete 34

8.3.6.9.6 Notification on update 34

8.3.7 Resource [*dmEventLog*] 34

8.3.7.1 Introduction 34

8.3.7.2 Create 34

8.3.7.3 Retrieve 34

8.3.7.4 Update 34

8.3.7.5 Delete 34

8.3.7.6 Notification on update 34

8.3.7.7 Resource [*retrieveLog*] 35

8.3.7.7.1 Introduction 35

8.3.7.7.2 Create 35

8.3.7.7.3 Retrieve 35

8.3.7.7.4 Update 35

8.3.7.7.5 Delete 35

8.3.7.7.6 Notification on update 35

8.3.8 Resource [*dmCapability*] 35

8.3.8.1 Introduction 35

8.3.8.2 Create 36

8.3.8.3 Retrieve 36

8.3.8.4 Update 36

8.3.8.5 Delete 36

8.3.8.6 Notification on update 36

8.3.8.7 Resource [*enable*] 36

8.3.8.7.1 Introduction 36

8.3.8.7.2 Create 36

8.3.8.7.3 Retrieve 36

8.3.8.7.4 Update 36

8.3.8.7.5 Delete 37

8.3.8.7.6 Notification on update 37

8.3.8.8 Resource [*disable*] 37

8.3.8.8.1 Introduction 37

8.3.8.8.2 Create 37

8.3.8.8.3 Retrieve 37

8.3.8.8.4 Update 37

8.3.8.8.5 Delete 37

8.3.8.8.6 Notification on update 37

8.3.9 Resource [*dmStorage*] 37

8.3.9.1 Introduction 37

8.3.9.2 Create 38

8.3.9.3 Retrieve 38

8.3.9.4 Update 38

8.3.9.5 Delete 38

8.3.9.6 Notification on update 38

8.3.9.7 Resource [*format*] 38

8.3.9.7.1 Introduction 38

8.3.9.7.2 Create 38

8.3.9.7.3 Retrieve 38

8.3.9.7.4 Update 39

8.3.9.7.5 Delete 39

8.3.9.7.6 Notification on update 39

8.3.9.8 Resource [*unmount*] 39

8.3.9.8.1 Introduction 39

8.3.9.8.2 Create 39

8.3.9.8.3 Retrieve 39

8.3.9.8.4 Update 39

8.3.9.8.5 Delete 39

8.3.9.8.6 Notification on update 39

8.3.10 Resource [*battery*] 40

8.3.10.1 Introduction 40

8.3.10.2 Create 40

8.3.10.3 Retrieve 40

8.3.10.4 Update 40

8.3.10.5 Delete 40

8.3.10.6 Notification on update 40

# 1 Scope

The present document defines general guidelines when interworking between external Proximal IoT technologies which are not aware of oneM2M-defined functionality, and the oneM2M system (i.e. the interaction between non-oneM2M-aware devices, gateways or applications (non-oneM2M entities) and oneM2M-defined entities). In the present document guidelines are defined on how to use oneM2M-defined resources to represent the state, events, actions, procedures, services provided by the non-oneM2M entities and how to expose oneM2M functions or services represented by oneM2M-defined resource to non-oneM2M Proximal IoT technologies. Therefore, services provided by non-oneM2M entities can be consumed by oneM2M entities via the oneM2M defined interfaces and vice versa. When following these guidelines, oneM2M-aware entities consuming services provided by non-oneM2M-aware entities via the specified interworking methods do not need to know anything about external Proximal IoT technologies. Also entities in an external Proximal IoT network that are not oneM2M-aware can consume services provided by oneM2M entities when exposed to the external Proximal IoT network according to the specified methods.

# 2 References

## 2.1 Normative references

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

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

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

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

[3] oneM2M TS-0023: " SDT based Information Model and Mapping for Vertical Industries ".

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

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

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

[7] oneM2M TS-0002: "Requirements".

[8] oneM2M TS-0004: "Service Layer Core Protocol".

## 2.2 Informative references

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

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

[i.1] oneM2M Drafting Rules.

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

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in oneM2M TS-0011 [1], oneM2M TS‑0002 [7] and the following apply:

NOTE: A term defined in the present document takes precedence over the definition of the same term, if any, in oneM2M TS-0011 [1] and oneM2M TS-0001 [2].

**proximal IoT:** IoT components communicating with each other directly in a local network using specific communication protocols and information models

NOTE 1: The notion of "proximal" is motivated by the fact that many of such IoT technologies are based on discovery and advertisement techniques that are designed to be used primarily in networks where all communicating entities are in close proximity with each other, In the current context "proximal" does not imply spatial proximity rather than being part of the same IoT network that is in general not using any oneM2M-defined-functionality.

NOTE 2: When not stated otherwise, entities in a Proximal IoT network are not aware of any oneM2M-defined functions or procedures.

**proximal IoT interworking:** exchange of information and exposure/consumption of services across the borders between entities designed for non-oneM2M-defined Proximal IoT technologies and oneM2M entities

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in oneM2M TS-0011 [1] and the following apply:

IoT Internet of Things

SDT Smart DeviceTemplate

DM SDT modules <flexContainer> specializations of SDT moduleClasses for Device Management, defined in oneM2M TS-0023 [3] clause 5.8.4 Conventions

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

# 5 Introduction

The scope of Proximal IoT Interworking is to enable the exchange of information between different things, devices and applications and the use of services they provide, irrespective of whether they are designed as oneM2M-defined entities according to the Functional architecture specified in oneM2M TS-0001 [2] or according to other non-oneM2M-defined Proximal IoT technologies. Proximal IoT Interworking can be modelled to be composed of actions on several layers: On the connection layer, on the resource framework layer and on the information model layer:

* Interworking on the connection layer - focus on the connection of entities. Two entities are interworkable if they support the same communication interface and communication protocol. Examples include Wifi connection, 3GPP wireless connection, etc. If two entities are interworkable on the connection layer, it is only guaranteed that data could be sent from one to another.
* Interworking on the resource framework layer - focus on the data types, resource template and data schemas. Two entities are interworkable if they share the same serializations, data types and resource templates. For example, if both entity can share information with the common understanding of xml schema, each entity will be able to recover the complete information contained in the message. Examples include SOAP, REST API, with specified serializations, etc.
* Interworking on the information model layer - focus on the information model, data model and common semantic understanding. Two entities are interworkable if they share the same information model and semantics. For example, in a smart home scenario, a light switch, a home gateway and an application that share the same information model can actually deploy the service of switching on and switching off the light if all of them use an information element with content "ON" to represent switching on the light and "OFF" to represent switching off the light. If the light switch is using "ON" but the application is using "TRUE" , the service cannot be deployed.

Interworking on the resource framework layer depends on the connection layer, and interworking on the information model layer depends on the resource framework layer.

To enable such consistent exchanges, oneM2M has designed the entire end to end architecture spanning entities for the platform (IN-CSE), gateways (MN-CSE) to devices (ASN and ADN), as described in oneM2M TS-0001 [2]. Corresponding to each layer, oneM2M has specified dedicated definitions for the enablement of:

* Interworking on the connection layer - Bindings defined by oneM2M i.e. HTTP, CoAP, MQTT and Websocket binding and associated procedures.
* Interworking on the resource framework layer - Serializations and resource structures defined by oneM2M.
* Interworking on the information model layer - The definition or the import of existing information models including the associated procedures in oneM2M, for example the SDT-based Information Model and Mapping for Vertical Industries in oneM2M TS‑0023 [3]. For device management purposes, it is either possible to use specializations of <mgmtObj> in various Technical Specifications oneM2M TS‑0001 [2], oneM2M TS-0022 [4], or to use SDT-based specializations of <flexContainer> that map moduleClasses specified in TS-0023 [3] clause 5.8.

The focus of the present document is the interworking on the information model layer and the implications on how to represent external Proximal IoT functions with means of resource instances in the oneM2M system.

However, the set of resource structures defined by oneM2M is very loosely coupled with the service of devices which may still cause interworking problems. Using CRUDN operations [2] on resources defined by oneM2M is the mechanism to enforce the common services oneM2M is trying to deliver. How to use these common services relies on interpretation of the implementer of the standard. For devices designed for non-oneM2M Proximal IoT technologies, if services of these devices are exposed to oneM2M entities using resources in inconsistent ways, it is still very hard to enable the interworking with these devices, because consumers of the services may need additional adaptation depending on different interpretations of resource content and relationships in different implementations.

In the present document, a general interworking architecture and framework to enable interworking up to the information model layer is defined.

For Device Management purposes, some generic guidelines for CRUDN operations on DM SDT modules are defined in clause 8. These guidelines can be refined in oneM2M Technical Specifications that detail the interworking with specific Proximal IoT Technologies.

# 6 General interworking architecture

## 6.1 Concept of Representation

In the oneM2M system, any kind of device, application or service or more generally speaking any kind of function that needs to be exposed to oneM2M-specified entities such as CSEs or AEs is represented by instances of specified resource types. Interaction with these termed functions is enabled by executing operations (e.g. create, delete, retrieve, update) on the resource instances. For example, by updating an attribute of a particular resource instance, an AE can change the state of a light actor from on to off. In addition, the concept of subscribing to resource instances and receiving notifications about content changes is also specified in oneM2M to allow for efficient monitoring of resource instances and thus the exposed function(s).

Following that fundamental concept of representing all functions by resource instances and their operations, real world devices or things in Proximal IoT networks that were not designed according to oneM2M specifications including the applications running on devices and/or services provided by devices can be represented by resource instances in the oneM2M system if interworking with those devices or things is needed. The resource instances are digital representations for the real world devices or things that are exposed to other entities in the oneM2M system via the oneM2M-specified interfaces for executing operations on those resource instances. How such resource instances, representing non-oneM2M Proximal IoT functions, are to be created and managed is described in the remainder of the present document. Furthermore, it is also described how to expose functions that are natively accessible in the oneM2M system, via the oneM2M-specified resource types and interfaces, to non-oneM2M Proximal IoT networks.

A representation of a non-oneM2M Proximal IoT function in a oneM2M-specified resource instance needs to be synchronized with the entity that it actually represents. For instance, if an attribute of a resource instance that represents the power state of an external Proximal IoT light is modified, the actual light state needs to be modified accordingly. The remainder of the present document describes how such synchronization is accomplished. Essentially any request targeting the representation of a non-oneM2M Proximal IoT function results in the corresponding interaction with the external device or thing in the Proximal IoT network. Similarly, any action in a non-oneM2M Proximal IoT function that is associated with a representation in the oneM2M system contained in a oneM2M resource instance shall result in an according operation on that resource instance.

Adhering to this concept of representation, oneM2M entities (AEs and CSEs) are enabled to request services or information provided by external non-oneM2M devices or things - termed "non-oneM2M Proximal IoT functions" - in the same way as interaction with native oneM2M devices or things - termed native oneM2M functions - is done.

## 6.2 Role of IPE(s)

### 6.2.1 Fit with the Functional Architecture

oneM2M is using an Interworking Proxy Entity (IPE) to handle the interworking between the oneM2M system and external non-oneM2M Proximal IoT functions residing on Non-oneM2M Device Nodes (NoDNs). The functional relationship between a NoDN and the CSE that is supposed to host the interworking functionality is defined in the following, see also Annex F of oneM2M TS-0001 [2].



Figure 6.2.1-1: Interworking through IPE

An Interworking Proxy Entity is an AE that supports both, the oneM2M Mca reference point as well as the non‑oneM2M interface that the functions on one or more connected NoDN(s) require. The IPE is responsible for synchronizing the services provided by the non-oneM2M Proximal IoT functions on the NoDN with oneM2M resource instances and vice versa. The IPE can be deployed together with the CSE or NoDN, or the IPE can also be deployed separately. Deployed together means running in the same execution environment or running on the same operating system where in such case, the communication between the deployed entities doesn't involve remote communication through wired or wireless network. In this kind of deployment, the communication method for supporting communication between the functional entities depicted in Figure 6.2.1-1 shall be provided by the execution environment or the operating system, e.g. inter-process communications, function calls, service call-backs or message bus technologies.

The IPE is registered to the CSE that is meant to host the interworking functionality and it translates the functions or services provided or consumed by one or more NoDN(s) to or from content of resource instances that are hosted by CSEs in the oneM2M system. When such resource instances represent functions or services provided by NoDN(s) connected to a specific IPE, the Registrar CSE of that IPE shall host those resource instances. The IPE shall translate any occurrence of operations on the oneM2M resource instances into invocation of the corresponding non-oneM2M Proximal IoT functions provided by the connected NoDN(s) and vice versa when non-oneM2M Proximal IoT functions executed on any connected NoDN(s) need to be reflected by change(s) of content in the corresponding representation(s) in oneM2M resource instances.

NOTE: More than one IPE may be instantiated in order to support interworking with an external non-oneM2M Proximal IoT technology. Also, an IPE may instantiate one or more supporting AEs that are used to simplify correlation with AE aspects such as service subscription profiles, access control privileges, authentication, authorization, etc.

### 6.2.2 Exposure of Proximal IoT functions to the oneM2M System

The role of an IPE, when it comes to exposure of external non-oneM2M Proximal IoT functions to the oneM2M System, includes the creation, monitoring, modification (update/delete) of resource instances that are supposed to represent those external functions on its own Registrar CSE. This role also includes the following:

* The IPE needs to determine which non-oneM2M Proximal IoT functions need to be exposed (e.g. through provisioning, discovery, on-demand signalling, etc.) and detect dynamic changes of the set of the non‑oneM2M Proximal IoT functions to be exposed. On-demand discovery or change of exposure configurations may be triggered by other AEs/CSEs by modifying corresponding resource instances created by the IPE. A request to trigger discovery or to demand a change of the exposure configuration can be accomplished, for instance, via creation and monitoring of a <*container*> resource instance by the IPE, under which authorized AEs can create <*contentInstance*> resource instances, that indicate which action to take. Details of such a triggering mechanism are implementation dependent and will not be further specified in the present document.
* The IPE needs to handle creation/deletion of resource instances representing non-oneM2M Proximal IoT functions according to the - possibly dynamically changing - need to expose them to the oneM2M system using resource types that are independent of the external Proximal IoT technology.
* The IPE is responsible to modify the resource instances representing the non-oneM2M Proximal IoT functions according to any state changes occurring in the external Proximal IoT system. If the IPE cannot be synchronously notified of state changes occurring in the external Proximal IoT system, but can execute a synchronous read function on this system, the IPE can monitor the RETRIEVE requests on the resources representing non-oneM2M Proximal IoT functions by adding to them an attribute *resourceMappingRules* with a *retargetCriteria* “operations=RETRIEVE”: in this case, the RETRIEVE requests from an external entity will be retargeted to the IPE, which can then retarget them to the external IoT Proximal system.
* The IPE is responsible for monitoring relevant changes in the resource instances representing the non-oneM2M Proximal IoT functions and invocation of appropriate non-oneM2M Proximal IoT function(s) when any operation(s) meant to trigger the execution of that non-oneM2M Proximal IoT function(s) occur for those resource instances in the hosting CSE. This monitoring can be achieved either by creating under each such resource a <*subscription*> resource with *eventNotificationType* attribute set to “Blocking\_Update”, or by adding to these resources an attribute *resourceMappingRules* with a *retargetCriteria* “operations=UPDATE”.

The set of responsibilities of the IPE when exposing non-oneM2M Proximal IoT functions to the oneM2M system is summarized in Figure 6.2.2-1. The dashed boxes describe optional/alternative means to determine the set of exposed functions. Note that, in this Figure one IPE is responsible for all interworking actions. More than one IPE may be used to interwork with one particular Proximal IoT network. Also additional AEs may get instantiated by an IPE to support interworking, see clause 6.1. Details on the resource mapping are contained in clause 7.



Figure 6.2.2-1: Exposure of Proximal IoT functions to the oneM2M System

### 6.2.3 Exposure of native oneM2M functions to the Proximal IoT System

The role of an IPE, when it comes to exposure of native oneM2M functions (aspects of devices, applications, services) to an external non-oneM2M Proximal IoT network, includes the monitoring, modification (update/delete) of resource instances that are representing oneM2M-internal functions to be exposed to an external non-oneM2M Proximal IoT network. This role also includes the following:

* The IPE needs to determine which native oneM2M functions need to be exposed to the external non-oneM2M Proximal IoT network (e.g. through provisioning, resource instance discovery, on-demand signalling, etc.) and detect dynamic changes of the set of native oneM2M functions to be exposed. On-demand discovery or change of exposure configurations may be triggered by other AEs/CSEs by modifying corresponding resource instances created by the IPE. A request to trigger discovery or to demand a change of the exposure configuration can be accomplished, for instance, via creation and monitoring of a <*container*> resource instance by the IPE, under which authorized AEs can create <*contentInstance*> resource instances, that indicate which action to take. Details of such a triggering mechanism are implementation dependent and will not be further specified in the present document.
* The IPE needs to handle creation/deletion of functions in the non-oneM2M Proximal IoT network representing native oneM2M functions according to the - possibly dynamically changing - need to expose them to the non-oneM2M Proximal IoT network.
* The IPE is responsible to modify the resource instances representing exposed native oneM2M functions according to any events occurring in the external non-oneM2M Proximal IoT network that require state changes of the resource instances representing the exposed native oneM2M functions.
* The IPE is responsible for monitoring relevant changes in the resource instances representing exposed native oneM2M functions and invocation of the corresponding non-oneM2M Proximal IoT function(s) provided by the IPE when any operation(s) occur for those resource instances that require an indication of the changed state in the provided external functions.

The set of responsibilities when exposing native oneM2M functions to the external non-oneM2M Proximal IoT network is summarized in Figure 6.2.3-1. The dashed boxes or lines describe optional/alternative means to determine the set of exposed functions. Note that, in this Figure one IPE is responsible for all interworking actions. More than one IPE may be used to interwork with one particular Proximal IoT network. Also additional AEs may get instantiated by an IPE to support interworking, see clause 6.1. Details on the resource mapping are contained in clause 7.



Figure 6.2.3-1: Exposure of native oneM2M functions to the Proximal IoT System

# 7 Representation of non-oneM2M entities in Proximal IoT networks

## 7.1 Representation of non-oneM2M Proximal IoT Devices

From a device management perspective in oneM2M, a device is represented using a <*node*> resource.

All management related capabilities of a device are then represented using child resources of a <*node*> resource. This principle shall also be applied for non-oneM2M Proximal IoT devices (which are NoDNs), i.e. all aspects of device management of a device subject to device management methods defined in oneM2M should be exposed by using child resources of a <*node*> resource. The <*node*> resource instances representing device management aspects of non-oneM2M Proximal IoT devices shall be created by the responsible IPE on the IPE's Registrar CSE.

Up to version 3.0.0 of this specification, valid for releases up to 4 of oneM2M, Device Management capabilities of a device were represented as <*mgmtObj*> child resources of the <*node*> resource. The operations on these <*mgmtObj*> resources are described in clause 10.2.8 of oneM2M TS-0001 [2] and Annex D of oneM2M TS-0004 [8]. From this version, this approach is deprecated and the preferred approach is to use, for device management operations, the DM SDT modules defined in clause 5.8 of oneM2M TS-0023 [3]. For this, the IPE shall create on its registrar CSE a [*flexNode*] <*flexContainer*> specialization as child of the <*node*> resource, and expose the device management capabilities of the device as DM SDT modules, children of the [*flexNode*] resource. The operations on these DM SDT modules are presented in clause 8.

If the device complies to oneM2M-defined information models - such as the ones defined in oneM2M TS-0023 [3] - the device should be represented using the respective specializations of resources specified in oneM2M. For example, a home domain device for a light as defined in oneM2M TS-0023 [3] is represented using the corresponding specializations of <*flexContainer*> defined by the [*deviceLight*] resource type. If the information model of the device is not defined by oneM2M, a <*flexContainer*> may also be used with its *containerDefinition* attribute configured with a URI linking to the schema definition for that device type specified by the respective organization. Also if the <flexContainer> resource represents a non-oneM2M Proximal IoT device, the resource may be linked with the corresponding <*node*> resource that is used to reflect device management aspects of the device or to indicate relationship(s) to applications on the device represented by <*AE*> resource instances, if applicable. The instances of specializations of <*flexContainer*> resource types representing non-oneM2M Proximal IoT devices shall be created by the responsible IPE on the IPE's Registrar CSE. The preferred parent resource for such specialization of <*flexContainer*> resource instances is the IPE's own <*AE*> resource instance. The linkage between an instance of a specialization of the <*flexContainer*> resource type, representing a non-oneM2M Proximal IoT device, and the corresponding <*node*> resource instance, that is used to reflect device management aspects or relationships to applications of the device, shall be established as follows:

1. If present, a *nodeLink* attribute of the <*flexContainer*> specialization instance, representing the non-oneM2M Proximal IoT device, shall point to the <*node*> resource instance.
2. Otherwise, a *mgmtLink* attribute of the <*flexContainer*> specialization instance, representing the non-oneM2M Proximal IoT device, shall point to a <*deviceInfo*> resource instance that is a child of the <*node*> resource instance.

For devices that do not follow any standardized information model nor have any management requirements, there is no distinct resource types to be instantiated in the oneM2M system for the representation of the device.

## 7.2 Representation of non-oneM2M Proximal IoT Applications

A general pattern in oneM2M is the use of instances of an <*AE*> resource to represent applications running on devices/gateways if needed. In addition, all services provided by these applications should be represented as child resources of the respective <*AE*> resource instance. By browsing the child resources of an <*AE*> resource instance, it is easily understood what are all the services provided by the respective application. If an application de-registers (i.e. the <*AE*> resource instance is deleted) from the system, all the resources representing its services are also deleted since they are child resources of the <*AE*> instance. For example, if an application is to report temperature data, after registration, an <*AE*> resource instance representing the application is created on the registrar CSE. The data reporting service is then e.g. represented as a <*container*> or *<flexContainer>* child resource of the <*AE*> resource. If the <*AE*> resource gets deregistered, the <*container*> or *<flexContainer>* resource is deleted at the same time. The same principles should be applied to represent non-oneM2M Proximal IoT Applications on NoDN(s) and the services they provide (see clause 7.3).

According to the specific needs in a service deployment, the service provider may deploy one or multiple application instances on one device. Each NoDN application instance that needs to be exposed to the oneM2M system shall be represented by one <*AE*> resource instance and be assigned with one unique AE-ID to identify the application instance.

Depending on the type of oneM2M node hosting an application, its <*AE*> resource instance(s) may be hosted by different types of CSEs in the oneM2M system:

* For applications on ASNs and MNs, the corresponding <*AE*> resources shall be created under the <CSEBase> of the corresponding ASN-CSE and MN-CSE, accordingly.
* For applications on an ADN, the corresponding <*AE*> resources shall be created under the <CSEBase> of the Registrar CSE of the ADN (which may be an MN-CSE or IN-CSE).

If there is a need to represent applications on interworked NoDNs - which is the case if the interworked applications need to be identifiable for the purpose of service subscription, charging, differentiation during access control enforcement, authentication, App-ID registry, etc. - then one or more <*AE*> resource instances shall be created to represent those applications. The IPE is responsible to issue requests to the oneM2M system on behalf of the interworked applications by using the AE-ID of the created <*AE*> resources. Care should be taken for determining the number of necessary security associations for the created <*AE*> resources. If there is no need to represent different applications when interacting with functions on interworked NoDNs just one <*AE*> resource shall be created as the representation of the IPE that is responsible for accessing the NoDN services.

When a non-oneM2M Proximal IoT application running on a NoDN is represented by an <*AE*> resource instance and at the same time device management aspects or relationships to applications of that NoDN are represented by a <*node*> resource instance, the *nodeLink* attribute of that <*AE*> resource instance shall point to the <*node*> resource instance corresponding to that NoDN. Also the reverse linkage via the *hostedAELinks* attribute of the <*node*> resource shall be established.

## 7.3 Representation of non-oneM2M Proximal IoT Services

oneM2M defines different types of resources that may be used to represent services provided by a device. When representing non-oneM2M Proximal IoT services from interworked NoDN(s), proper resource types shall be chosen since the misusage of resource types for representing services may cause interoperability problems. General guidelines for resource representation of different services are as follows:

* For device management services: Up to version 3.0.0 of this document, specialized <*mgmtObj*> resource types as specified in oneM2M TS-0001 [2] and oneM2M TS-0022 [4], and <*mgmtCmd*>, *<execInstance>* as specified in oneM2M TS-0001 [2] shall be used. From version ???, specialized <*flexContainer*> resource types as specified in oneM2M TS-0023 [3], clause 5.8, shall be used. These resources shall be created by the responsible IPE as child resources of the <*node*> resource, or of the [*flexNode*] child of this <*node*>, which represents the managed device (see clause 7.1).
* Services defined in vertical domains specified in oneM2M TS-0023 [3] (agriculture, common, city, health, home, industry, railway, vehicular…): Specialized <*flexContainer*> resource types for moduleClasses as specified in oneM2M TS-0023 [3] shall be used to represent those services.
* Data management services (not covered by oneM2M TS-0023 [3]): <*container*>, <*contentInstance*>, <*timeSeries*>, <*timeSeriesInstance*> as specified in oneM2M TS-0001 shall be used.
* Location services: <*locationPolicy*>, <*container*>, <*contentInstance*>, <*latest*>, <*oldest*> as specified in oneM2M TS-0001 [2] shall be used.
* Group services: <*group*>, <*fanOutPoint*>, *<localMulticastGroup>* as specified in oneM2M TS-0001 [2] shall be used
* Event/notification services: <*subscription*>, <*notificationTargetSelfReference*>, <*notificationTargetMgmtPolicyRef*>, <*notificationTargetPolicy*>, <*policyDeletionRules*> as specified in oneM2M TS-0001 [2] shall be used.
* Security services: <*accessControlPolicy*>, <*dynamicAuthorizationConsultation*>, <*role*>, <*token*>, <*authorizationDecision>, <authorizationPolicy>, <authorizationInformation>* as specified in oneM2M TS‑0001 [2] and oneM2M TS-0003 [5] shall be used
* Semantic services: <*semanticDescriptor*>, <*ontologyRepository*>, <*ontology*>, <*semanticValidation*>, <*semanticMashupJobProfile*>, <*semanticMashupInstance*>, *<mashup>, <semanticMashupResult>* as specified in oneM2M TS-0001 [2] and oneM2M TS-0034 [6] shall be used.
* Charging services: *<statsConfig>, <eventConfig>, <statsCollect>* as specified in oneM2M TS-0001 [2] shall be used.

There are two ways of expressing relationships between resources as well as relationships between the services these resources represent: Parent-child relationship and linkage relationship. The linkage relationship only applies to specific oneM2M resource types such as <*accessControlPolicy*>, announced resources, and <*mgmtObj*> resources, etc.

The parent-child relationship of resources shall be used when the service represented by the child resource cannot exist independent of the services represented by the parent resource. If the parent service is deleted, the child services shall be deleted automatically.

# 8 Device Management Operations

This clause specifies the procedures for managing device capabilities, using SDT DM <flexContainer> specializations. This is an alternative to the approach based on <mgmtObj> or <mgmtCmd> resources specified in oneM2M TS-0001 [2] clause 10.2.8.

This clause describes the management procedures over Mca and Mcc reference points. The [flexNode] and SDT DM *<flexContainer>* resources are hosted on the CSE of the managed entity when the managed entity is an ASN, MN or IN. If the managed entity is an ADN node or the managed entity is co-located on an ASN, MN or IN, the [flexNode] and SDT DM *<flexContainer*> resources are hosted on the registrar CSE of the managed entity. The DM *<flexContainer>*, its parent [flexNode] and its grand-parent *<node>* resources hosted on node's CSE may be announced to associated IN-CSEs.

In the scenario where the managed entity is a NoDN, the managed entities' DM *<flexContainer>,* its parent [flexNode] and its grand-parent *<node>* resources are hosted on the registrar CSE of the IPE that manages this entity.

The Node management, as described in oneM2M TS-0001 [2] clauses 10.2.8.2 to 10.2.8.6, is unchanged, but in this case the only child of the <node> resource will be a [flexNode] specialization.

## 8.1 [*flexNode*] management

### 8.1.0 Access Control

The SDT DM resources, children of the [*flexNode*], should only be created by the the IPE that manages them, or at least by an AE that is referenced in the *hostedAELinks* of the parent <*node*> resource. This could be achieved by the creation of an <*accessControlPolicy*> that forbids the creation and deletion of <*flexContainer*> resources to any originator other than the entities defined in the parent <*node*>’s *hostedAELinks* attribute.

Any DM SDT <*flexContainer*> child of [*flexNode*] that would not be created by such an AE would not be managed by the IPE.

### 8.1.1 Create [*flexNode*]

This procedure shall be used for creating a *[flexNode]* resource.

Table 8.1.1-1: *[flexNode]* CREATE

| ***[flexNode]* CREATE** |
| --- |
| Information in Request message | All parameters defined in TS-0001 table 8.1.2-3 apply with the specific details for:***From:*** Identifier of the IPE that initiates the Request***To:*** The address of the *<node>* where the *[flexNode]* resource is intended to be Created***Content:*** The representation of the [*flexNode*] resource described in clause 5.8.2 in TS-0023. The accessControlPolicyIDs will contain the identifiers of the <accessControlPolicy> resources described in clause 8.1.0. |
| Processing at Originator before sending Request | According to clause TS-0001 10.1.2.The Originator shall be an IPE that manages the corresponding entity in the Proximal IoT System. |
| Processing at Receiver | According to clause TS-0001 10.1.2 |
| Information in Response message | All parameters defined in TS-0001 table 8.1.3-1 apply with the specific details for:* ***Content*:** Address of the created *[flexNode]* resource, according to clause TS-0001 10.1.2
 |
| Processing at Originator after receiving Response | According to clause TS-0001 10.1.2 |
| Exceptions | According to clause TS-0001 10.1.2 |

#### 8.1.1.1 Create <*subscription*>

The IPE shall create a <*subscription*> resource, child of the [flexNode] resource, with the attributes given in Table 8.1.1-2.

Table 8.1.1.1-1: <*subscription*> resource

| Attributes of *<subscription> resource* | Description / Value  |
| --- | --- |
| *notificationURI* | IPE URI  |
| *notificationContentType* | all attributes |
| *notificationEventType* | D. Deletion of a direct child of the subscribed-to resource.G. Update to attributes of thesubscribed-to resource with blocking UPDATE |

#### 8.1.1.2 Create DM SDT modules

As specified in clause 5.8.2 of TS-0023, the ‘flexNode’ has at least as mandatory child a ‘dmDeviceInfo’ ModuleClass The IPE, after creating the [*flexNode*] resource, shall therefore create a [*dmDeviceInfo*] <*flexContainer*> specialization under the [*flexNode*].

It also shall create the <*flexContainer*> specializations that correspond to the DM SDT modules that represent the device management capabilities of the Proximal IoT devices it manages.

### 8.1.2 Retrieve [*flexNode*]

This procedure shall be used for retrieving the attributes of a *[flexNode]* resource.

Table 8.1.2-1: *[flexNode]* RETRIEVE

|  |
| --- |
| *[flexNode]* RETRIEVE |
| Information in Request message | All parameters defined in TS-0001 table 8.1.2-3 apply with the specific details for:***Content*:** Void |
| Processing at Originator before sending Request | According to clause TS-0001 10.1.3 |
| Processing at Receiver | According to clause TS-0001 10.1.3 |
| Information in Response message | All parameters defined in TS-0001 table 8.1.3-1 apply with the specific details for:***Content*:** Attributes of the *[flexNode]* resource as defined in clause TS-0023 5.8.2. |
| Processing at Originator after receiving Response | According to clause TS-0001 10.1.3 |
| Exceptions | According to clause TS-0001 10.1.3 |

### 8.1.3 Update [*flexNode*]

This procedure shall be used for updating the attributes and the actual data of a [*flexNode*] resource and its child resources.

Table 8.1.3-1: *[flexNode]* UPDATE

| *[flexNode]* UPDATE |
| --- |
| Information in Request message | All parameters defined in TS-0001 table 8.1.2-3 apply with the specific details for:***Content***: attributes of the *[flexNode]* resource as defined in clause TS-0023 5.8.2 which need be updated, with the exception of the Read Only (RO) attributes that cannot be modified |
| Processing at Originator before sending Request | According to clause TS-0001 10.1.4 |
| Processing at Receiver | According to clause TS-0001 10.1.4 with the following:* The Receiver shall check whether the provided attributes of the [flexNode] resource represent a valid request for updating *[flexNode]* resource
 |
| Information in Response message | According to clause TS-0001 10.1.4 |
| Processing at Originator after receiving Response | According to clause TS-0001 10.1.4 |
| Exceptions | According to clause TS-0001 10.1.4 |

### 8.1.4 Delete [*flexNode*]

This procedure shall be used for deleting an existing *[flexNode]* resource.

NOTE: The deletion of the *[flexNode]* resource is on discretion of the Originator IPE.

Table 8.1.4-1: *[flexNode]* DELETE

| *[flexNode]* DELETE |
| --- |
| Information in Request message | All parameters defined in TS-0001 table 8.1.2-3 apply |
| Processing at Originator before sending Request | According to clause TS-0001 10.1.5The Originator shall be an IPE that manages the corresponding entity in the Proximal IoT System |
| Processing at Receiver | According to clause TS-0001 10.1.5 |
| Information in Response message | According to clause TS-0001 10.1.5 |
| Processing at Originator after receiving Response | According to clause TS-0001 10.1.5 |
| Exceptions | According to clause TS-0001 10.1.5 |

## 8.2 Generic DM SDT modules management

Device Management moduleClasses defined in TS-0023 [3] clause 5.8 are mapped as <*flexContainer*> specializations. These resources are hosted on the CSE of the managed entity when the managed entity is an ASN, MN or IN. If the managed entity is an ADN node or the managed entity is co-located on an ASN, MN or IN, the DM SDT *<flexContainer*> resources are hosted on the registrar CSE of the managed entity. If the managed entory is a NoDN node, the resources are hosted on the Registrar CSE of the IPE that manages them, The DM SDT *<flexContainer*> resource, its parent [*flexNode*] resource and its grand-parent *<node>* resource hosted on node's CSE may be announced to associated IN-CSEs.

NOTE: This clause defines generic guidelines for handling all DM SDT modules; clause 8.3. will add detailed information for each specific module.

### 8.2.1 Create DM SDT *<flexContainer>*

Besides the generic create procedure defined in oneMEM TS-0001 clause 10.1.2, the procedure in table 8.2.1-1shall be used.

Table 8.2.1-1: DM SDT *<flexContainer>* CREATE

|  |
| --- |
| *<flexContainer>* CREATE  |
| Information in Request message | ***From:*** Identifier of the IPE that initiates the Request***To:*** The address of the *[flexNode]* where the *<flexContainer>* resource is intended to be Created***Content:*** The representation of the *<flexContainer>* resource for which the attributes are described in oneM2M TS-0023 clause 5.8. |
| Processing at Originator before sending Request | The Originator shall be an IPE that manages the corresponding entity in the Proximal IoT Technology.The Originator firstcollects the Proximal IoT Technology specific data model object (the management tree structure or also the value of the tree nodes if needed) of the Proximal IoT device and transforms the object into the DM SDT *<flexContainer>* resource representation, then requests the Hosting CSE to create the corresponding *<flexContainer>* resource. |
| Processing at Receiver | The Hosting CSE shall check that the request’s originatorID is contained in the *hostedAELinks* attribute of the grand-parent <node> resource. |
| Information in Response message | According to clause 10.1.2 |
| Processing at Originator after receiving Response | None |
| Exceptions | * The creation of the DM SDT *<flexContainer>* object is not allowed
* The created DM SDT *<flexContainer>* object already exists
 |
|  |

#### 8.2.1.1 Create <*subscription*>

Update requests to this created <*flexContainer*> shall be retargeted to the IPE that created it. For this, the IPE shall create a <*subscription*> resource, child of the <*flexContainer*> resource, with the attributes given in Table 8.2.1-2.

Table 8.2.1.1-1: <*subscription*> resource

| Attributes of *<subscription> resource* | Description / Value  |
| --- | --- |
| *notificationURI* | IPE URI  |
| *notificationContentType* | all attributes |
| *notificationEventType* | D. Deletion of a direct child of the subscribed-to resource.G. Update to attributes of thesubscribed-to resource with blocking UPDATE |

### 8.2.2 Retrieve DM SDT *<flexContainer>*

This procedure shall be used to retrieve information from an existing DM SDT *<flexContainer>* resource. Besides the generic retrieve procedure defined in oneM2M TS-0001 clause 10.1.3, the procedure in table 8.2.2-1 shall be used.

 Table 8.2.2-1: DM SDT *<flexContainer >* RETRIEVE

|  |
| --- |
| *<flexContainer>* RETRIEVE  |
| Information in Request message | ***From:*** Identifier of the AE or the CSE that initiates the Request***To:*** The address of the DM SDT *<flexContainer >* resource |
| Processing at Originator before sending Request | None |
| Processing at Receiver | According to clause 10.1.3,  |
| Information in Response message | Error code if the new technology specific data model object cannot be retrieved |
| Processing at Originator after receiving Response | None |
| Exceptions | * Corresponding technology specific object data cannot be retrieved from the managed entity (e.g. technology specific object not found)
 |

### 8.2.3 Update DM SDT *<flexContainer >*

This procedure shall be used to update information of an existing DM SDT *<flexContainer >* resource. Besides the generic update procedure defined in oneM2M TS-0001 clause 10.1.4, the procedure in table 8.2.3-1 shall be used.

Table 8.2.3-1: *<flexContainer>* UPDATE

|  |
| --- |
|  **DM SDT *<flexContainer >* UPDATE**  |
| Information in Request message | ***From:*** Identifier of the AE or the CSE that initiates the Request***To:*** The address of the DM SDT *<flexContainer >* resource***Content:*** The representation of the *<flexContainer >* resource for which the attributes are described in OneM2M TS-0023 clause 5.8. |
| Processing at Originator before sending Request | None |
| Processing at Receiver | Because of the <subscription> child of this <flexContainer> resource (see clause 8.2.1.1), this request will be retargeted to the managing IPE (see clause 8.2.5). |
| Information in Response message | Error code if a timeout happens before the retargeting completes, otherwise returns the retargeted response. |
| Processing at Originator after receiving Response | None |
| Exceptions |  |

NOTE: The creation and update of the attributes that correspond to datapoints specified as R (read only) in clauses 5.8.x of TS-0023 are allowed only to the Originator IPE. The datapoints specified as RW are updatable by external entities. This is ensured by the IPE when it receives the retargeted UPDATE request.

### 8.2.4 Delete DM SDT *<flexContainer >*

This procedure shall be used to delete an existing DM SDT *<flexContainer >* resource. Besides the generic delete procedure defined in oneM2M TS-0001 clause 10.1.5, the procedure in table 8.2.4-1 shall be used.

Table 8.2.4-1: DM SDT *<flexContainer >* DELETE

|  |
| --- |
| **DM SDT *<flexContainer >* DELETE** |
| Information in Request message | ***From:*** Identifier of the AE, or the CSE that initiates the Request***To:*** The address of the *<flexContainer>* resource |
| Processing at Originator before sending Request |  |
| Processing at Receiver | For the DELETE operation, besides the common create operation defined in clause 10.1.5, the Receiver shall: |
| Information in Response message | Error code if the technology specific data model object cannot be deleted |
| Processing at Originator after receiving Response | None |
| Exceptions | * Corresponding technology specific data model object cannot be deleted from managed entity (e.g. not reachable, technology specific data model object not found)
 |
|  |

### 8.2.5 Notifications on *<flexContainer>* operations

Following the <*subscription*> resource defined in clause 8.2.1.1, the IPE shall be notified when a <*flexContainer*> specialization child of the [*flexNode*] resource, representing a DM SDT module, will be updated by an external entity. The IPE is responsible for invoking the appropriate non-oneM2M Proximal IoT function(s) when any change in the <*flexContainer*> resource, as specified in the received notification, is meant to trigger the execution of that non-oneM2M Proximal IoT function.

## 8.3 Specific DM SDT modules management

### 8.3.1 Resource [*dmDeviceInfo*]

#### 8.3.1.1 Introduction

The detailed description of the [*dmDeviceInfo*] resource can be found in clause 5.8.4 of the oneM2M TS-0023 [3].

Table 8.3.1.1‑1: Data Type Definition of [*dmDeviceInfo*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmDeviceInfo,dmDeviceInfoAnnc | MAD-mod-dmDeviceInfo-vx\_y\_z.xsd |  |

NOTES:

* The creation and deletion of the [*dmDeviceInfo*] resource is on discretion of the Originator IPE.
* As specified in clause 5.8.2 of TS-0023, the ‘flexNode’ has at least as mandatory child a ‘dmDeviceInfo’ ModuleClass. The IPE, after creating the [*flexNode*] resource, shall therefore create a [*dmDeviceInfo*] <*flexContainer*> specialization under the [*flexNode*].

#### 8.3.1.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the ‘dmDeviceInfo’ ModuleClass.

In particular, some datapoints should be filled, for instance *serialNumber*, *manufacturer* and *model* when this information is available.

#### 8.3.1.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.1.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.1.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.1.6 Notification on update

No change from the generic procedures in clause 8.2.5.

### 8.3.2 Resource [*dmAgent*]

#### 8.3.2.1 Introduction

The detailed description of the [*dmAgent*] resource can be found in clause 5.8.3 of the oneM2M TS-0023 [3].

Table 8.3.2.1‑1: Data Type Definition of [*dmAgent*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmAgent,dmAgentAnnc | MAD-mod-dmAgent-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmAgent*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology can provide at least a rebooting function on the corresponding Proximal IoT devices and read their status.

#### 8.3.2.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmAgent ModuleClass, at least the ‘state’ datapoint (attribute ‘*state*’).

As the ‘reboot’ SDT action is mandatory in clause TS-0023 5.8.3, the IPE shall create a <*flexContainer*> [*reboot*] specialization as child of the [*dmAgent*] resource.

If the Proximal IoT Technology allows deploying new packages on the Proximal IoT device, the IPE will create a <*flexContainer*> [*deployPackage*] specialization as child of the [*dmAgent*] resource.

#### 8.3.2.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.2.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.2.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.2.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.2.7 Resource [*reboot*]

##### 8.3.2.7.1 Introduction

The detailed description of the [*reboot*] resource can be found in clause 5.8.3 of the oneM2M TS-0023 [3].

Table 8.3.2.7.1‑1: Data Type Definition of [*reboot*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| reboot,rebootAnnc | MAD-act-reboot-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*reboot*] resource is on discretion of the Originator IPE. It is created as child of a [*dmAgent*] resource.

##### 8.3.2.7.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the argument ‘rebootType’ of the reboot SDT action as an attribute *rebTe* of the [*reboot*] resource.

##### 8.3.2.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.2.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.2.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.2.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a reboot of the corresponding Proximal IoT device, according to the rebootType argument.

The IPE shall update the parent *state* attribute according to the known status of the Proximal IoT device.

#### 8.3.2.8 Resource [*deployPackage*]

##### 8.3.2.8.1 Introduction

The detailed description of the [*deployPackage*] resource can be found in clause 5.8.3 of the oneM2M TS-0023 [3].

Table 8.3.2.8.1‑1: Data Type Definition of [deployPackage]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| deployPackage,deployPackageAnnc | MAD-act-deployPackage-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*deployPackage*] resource is on discretion of the Originator IPE. It is created as child of a [*dmAgent*] resource.

##### 8.3.2.8.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the argument ‘name’, ‘version’ and ‘url’ of the deployPackage SDT action as attributes *name, versn* and *url* of the [*deployPackage*] resource.

##### 8.3.2.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.2.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.2.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.2.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall create a [*dmPackage*] <*flexContainer*> resource as child of the grand-parent [*flexNode*] resource. It will create a *result* (short name *resut*) attribute of the [deployPackage] resource and fill it with the resourceID of the created [*dmPackage*] resource.

Note: the IPE shall not, at this stage, attempts to deploy the package on the Proximal IoT device: this is done by manipulating the created [*dmPackage*] resource.

### 8.3.3 Resource [*dmDataModelIO*]

#### 8.3.3.1 Introduction

The detailed description of the [*dmDataModelIO*] resource can be found in clause 5.8.5 of the oneM2M TS-0023 [3].

Table 8.3.3.1‑1: Data Type Definition of [*dmDataModelIO*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmDataModelIO,dmDataModelIOAnnc | MAD-mod-dmDataModelIO-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmDataModelIO*] resource is on discretion of the Originator IPE. It shall be created if the Proximal IoT Technology allows reading and/or writing parameters of the underlying data model.

#### 8.3.3.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall create the ‘dataModelType’ datapoint of the dmDataModelIO ModuleClass, as a *daMTe* attribute.

If the Proximal IoT Technology allows reading data model parameters, the IPE will create a <*flexContainer*> [*readIO*] specialization as child of the [*dmDataModelIO*] resource.

If the Proximal IoT Technology allows writing data model parameters, the IPE will create a <*flexContainer*> [*writeIO*] specialization as child of the [*dmDataModelIO*] resource.

#### 8.3.3.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.3.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.3.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.3.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.3.7 Resource [*readIO*]

##### 8.3.3.7.1 Introduction

The detailed description of the [*readIO*] resource can be found in clause 5.8.5 of the oneM2M TS-0023 [3].

Table 8.3.3.7.1‑1: Data Type Definition of [*readIO*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| readIO,readIOAnnc | MAD-act-readIO-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*readIO*] resource is on discretion of the Originator IPE. It is created as child of a [*dmDataModelIO*] resource.

##### 8.3.3.7.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the argument ‘address’ of the readIO SDT action as an attribute *addrs* of the [*readIO*] resource.

##### 8.3.3.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.3.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.3.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.3.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a read of the parameter(s) referenced by the *address* argument on the corresponding Proximal IoT device. It will create a *result* (short name *resut*) attribute of the [*dmDataModelIO*] resource and fill it with the values returned by the device.

#### 8.3.3.8 Resource [*writeIO*]

##### 8.3.3.8.1 Introduction

The detailed description of the [*writeIO*] resource can be found in clause 5.8.5 of the oneM2M TS-0023 [3].

Table 8.3.3.8.1‑1: Data Type Definition of [*readIO*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| writeIO,writeIOAnnc | MAD-act-writeIO-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*writeIO*] resource is on discretion of the Originator IPE. It is created as child of a [*dmDataModelIO*] resource.

##### 8.3.3.8.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the arguments ‘address’ and ‘payload’ of the writeIO SDT action as attribute *addrs* and *payld* of the [*writeIO*] resource.

##### 8.3.3.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.3.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.3.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.3.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a write of the parameter(s) referenced by the *address* argument, with values in the *payload* argument, on the Proximal IoT device. It will create a *result* (short name *resut*) attribute of the [*dmDataModelIO*] resource and fill it with the values returned by the device.

### 8.3.4 Resource [*dmFirmware*]

#### 8.3.4.1 Introduction

The detailed description of the [*dmFirmware*] resource can be found in clause 5.8.6 of the oneM2M TS-0023 [3].

Table 8.3.4.1‑1: Data Type Definition of [*dmFirmware*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmFirmware,dmFirmwareAnnc | MAD-mod-dmFirmware-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmFirmware*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows handling the firmware of the Proximal IoT devices.

#### 8.3.4.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmFirmware ModuleClass, at least the mandatory ones.

If the Proximal IoT Technology allows updating the firmware of a Proximal IoT device, the IPE will create a <*flexContainer*> [*updateFirmware*] specialization as child of the [*dmFirmware*] resource.

If the Proximal IoT Technology allows Proximal IoT devices to toggle between the installed firmware and a backup firmware, the IPE will create a <*flexContainer*> [*toggle*] specialization as child of the [*dmFirmware*] resource.

#### 8.3.4.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.4.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.4.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.4.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.4.7 Resource [*updateFirmware*]

##### 8.3.4.7.1 Introduction

The detailed description of the [*updateFirmware*] resource can be found in clause 5.8.6 of the oneM2M TS-0023 [3].

Table 8.3.4.7.1‑1: Data Type Definition of [*updateFirmware*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| updateFirmware,updateFirmware Annc | MAD-act-updateFirmware-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*updateFirmware*] resource is on discretion of the Originator IPE. It is created as child of a [*dmFirmware*] resource.

##### 8.3.4.7.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the arguments ‘url’ and ‘version’ of the updateFirmware SDT action as attribute *url* and *versn* of the [*updateFirmware*] resource.

##### 8.3.4.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.4.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.4.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.4.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger an update of a new firmware, referenced by the *url* and *version* argument, on the Proximal IoT device. It will create a *result* (short name *resut*) attribute of the [*updateFirmware*] resource and fill it with the message returned by the device.

Note: the steps of the installation (download of the firmware, installation, etc.) are left to the implementor IPE, but the IPE shall fill the *primaryState* attribute of the parent [*dmFirmware*] resource accordingly.

#### 8.3.4.8 Resource [*toggle*]

##### 8.3.4.8.1 Introduction

The detailed description of the [*toggle*] resource can be found in clause 5.8.6 of the oneM2M TS-0023 [3].

Table 8.3.4.8.1‑1: Data Type Definition of [*toggle*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| toggle,toggleAnnc | MAD-act-toggle-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*toggle*] resource is on discretion of the Originator IPE. It is created as child of a [*dmFirmware*] resource.

##### 8.3.4.8.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.4.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.4.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.4.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.4.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a toggle between the current firmware and a backup one, on the Proximal IoT device. It will create a *result* (short name *resut*) attribute of the [*toggle*] resource and fill it with the message returned by the device.

Note: the steps of the toggle (download of the firmware, installation, etc.) are left to the implementor IPE, but the IPE shall fill the *primaryState* attribute of the parent [*dmFirmware*] resource accordingly.

### 8.3.5 Resource [*dmSoftware*]

#### 8.3.5.1 Introduction

The detailed description of the [*dmSoftware*] resource can be found in clause 5.8.7 of the oneM2M TS-0023 [3].

Table 8.3.5.1‑1: Data Type Definition of [*dmSoftware*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmSoftware,dmSoftwareAnnc | MAD-mod-dmSoftware-vx\_y\_z.xsd |  |

NOTES:

* the creation and deletion of the [*dmSoftware*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows installing various software images on the Proximal IoT devices. The IPE shall create one [*dmSoftware*] resource per software image.
* a [*dmSoftware*] resource can be created either at the initialization if it represents a software module that is pre-installed on the device, or after installation of one or more [dmPackage] module(s) (see clause 8.3.6) that have been dynamically created (for instance a software image with associated configuration files and libraries). See TS-0023 [3] clause 5.8.7.

#### 8.3.5.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmSoftware ModuleClass, at least the mandatory ones.

If the Proximal IoT Technology allows activating/deactivating the software of a Proximal IoT device, the IPE will create <*flexContainer*> [*activate*] and or [*deactivate*] specialization as child of the [*dmSoftware*] resource.

#### 8.3.5.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.5.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.5.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.5.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.5.7 Resource [*activate*]

##### 8.3.5.7.1 Introduction

The detailed description of the [*activate*] resource can be found in clause 5.8.7 of the oneM2M TS-0023 [3].

Table 8.3.5.7.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| activate,activateAnnc | MAD-act-activate-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*activate*] resource is on discretion of the Originator IPE. It is created as child of a [*dmSoftware*] resource.

##### 8.3.5.7.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.5.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.5.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.5.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.5.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger an activation of the software on the Proximal IoT device.

Note: the steps of the activation are left to the implementor IPE, but the IPE shall fill the *state* attribute of the parent [*dmSoftware*] resource accordingly.

#### 8.3.5.8 Resource [*deactivate*]

##### 8.3.5.8.1 Introduction

The detailed description of the [*deactivate*] resource can be found in clause 5.8.7 of the oneM2M TS-0023 [3].

Table 8.3.5.8.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| deactivate,deactivateAnnc | MAD-act-deactivate-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*deactivate*] resource is on discretion of the Originator IPE. It is created as child of a [*dmSoftware*] resource.

##### 8.3.5.8.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.5.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.5.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.5.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.5.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a deactivation of the software on the Proximal IoT device.

Note: the steps of the deactivation are left to the implementor IPE, but the IPE shall fill the *state* attribute of the parent [*dmSoftware*] resource accordingly.

### 8.3.6 Resource [*dmPackage*]

#### 8.3.6.1 Introduction

The detailed description of the [*dmPackage*] resource can be found in clause 5.8.9 of the oneM2M TS-0023 [3].

Table 8.3.6.1‑1: Data Type Definition of [*dmPackage*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmPackage,dmPackageAnnc | MAD-mod-dmPackage-vx\_y\_z.xsd |  |

NOTES:

* the creation and deletion of the [*dmPackage*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows installing various packages (libraries, configuration files, softwarevimages, etc) on the Proximal IoT devices. The IPE shall create one [*dmPackage*] resource per package.
* a [dmPackage] resource can be created either at the initialization if it represents a package that is pre-installed on the device, or by a deployment from the [*dmAgent*] module (see clause 8.3.2). See TS-0023 [3] clause 5.8.9.

#### 8.3.6.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmPackage ModuleClass, at least the mandatory ones.

As the ‘install’, ‘uninstall’ and ‘update’ SDT actions are mandatory in clause TS-0023 5.8.9, the IPE shall create <*flexContainer*> [*install*], [*uninstall*], [*update*] specializations as children of the [*dmPackage*] resource.

#### 8.3.6.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.6.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.6.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.6.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.6.7 Resource [*install*]

##### 8.3.6.7.1 Introduction

The detailed description of the [*install*] resource can be found in clause 5.8.9 of the oneM2M TS-0023 [3].

Table 8.3.6.7.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| install,installAnnc | MAD-act-install-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*install*] resource is on discretion of the Originator IPE. It is created as child of a [*dmPackage*] resource.

##### 8.3.6.7.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.6.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.6.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.6.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.6.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger an installation of the package on the Proximal IoT device.

Note: the steps of the installation (download, install, etc.) are left to the implementor IPE, but the IPE shall fill the *state* attribute of the parent [*dmPackage*] resource accordingly.

#### 8.3.6.8 Resource [*uninstall*]

##### 8.3.6.8.1 Introduction

The detailed description of the [*uninstall*] resource can be found in clause 5.8.9 of the oneM2M TS-0023 [3].

Table 8.3.6.8.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| uninstall,uninstallAnnc | MAD-act-uninstall-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*uninstall*] resource is on discretion of the Originator IPE. It is created as child of a [*dmPackage*] resource.

##### 8.3.6.8.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.6.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.6.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.6.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.6.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a deinstallation of the package on the Proximal IoT device.

Note: the steps of the deinstallation are left to the implementor IPE, but the IPE shall fill the *state* attribute of the parent [*dmPackage*] resource accordingly.

#### 8.3.6.9 Resource [*update*]

##### 8.3.6.9.1 Introduction

The detailed description of the [*update*] resource can be found in clause 5.8.9 of the oneM2M TS-0023 [3].

Table 8.3.6.9.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| update,updateAnnc | MAD-act-update-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*update*] resource is on discretion of the Originator IPE. It is created as child of a [*dmPackage*] resource.

##### 8.3.6.9.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the arguments ‘url’ and ‘version’ of the update SDT action as attribute *url* and *versn* of the [*update*] resource.

##### 8.3.6.9.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.6.9.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.6.9.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.6.9.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger an update of the package on the Proximal IoT device.

Note: the steps of the update (download, install, etc.) are left to the implementor IPE, but the IPE shall fill the *state* attribute of the parent [*dmPackage*] resource accordingly.

### 8.3.7 Resource [*dmEventLog*]

#### 8.3.7.1 Introduction

The detailed description of the [*dmEventLog*] resource can be found in clause 5.8.8 of the oneM2M TS-0023 [3].

Table 8.3.7.1‑1: Data Type Definition of [*dmEventLog*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmEventLog,dmEventLogAnnc | MAD-mod-dmEventLog-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmEventLog*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows monitoring the logs on the Proximal IoT devices. The IPE shall create one [*dmEventLog*] resource per type of log (see enumeration logTypeId).

#### 8.3.7.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmEventLog ModuleClass, at least the mandatory ones.

If the Proximal IoT Technology allows retrieving the logs of a Proximal IoT device, the IPE will create a <*flexContainer*> [*retrieveLog*] specialization as child of the [*dmEventLog*] resource.

#### 8.3.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.7.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.7.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

When the *enabled* attribute is set to TRUE (resp. FALSE), the IPE shall trigger a start (resp. a stop) of the logging behaviour on the Proximal IoT device. It shall modify the *status* attribute accordingly.

#### 8.3.7.7 Resource [*retrieveLog*]

##### 8.3.7.7.1 Introduction

The detailed description of the [*retrieveLog*] resource can be found in clause 5.8.9 of the oneM2M TS-0023 [3].

Table 8.3.7.7.1‑1: Data Type Definition of [*activate*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| retrieveLog,retrieveLogAnnc | MAD-act-retrieveLog-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*retrieveLog*] resource is on discretion of the Originator IPE. It is created as child of a [*dmEventLog*] resource.

##### 8.3.7.7.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent the arguments ‘start’ and ‘end’ of the update SDT action as attribute *start* and *end* of the [*retrieveLog*] resource.

##### 8.3.7.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.7.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.7.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.7.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall trigger a retrieval of the logs of the Proximal IoT device.

Note: the steps of the retrieval are left to the implementor IPE, but the IPE shall fill the *status* attribute of the parent [*dmEventLog*] resource accordingly.

When retrieval is completed, the IPE shall fill the *data* attribute of the prent [dmEventLog] with the retrieved log value.

### 8.3.8 Resource [*dmCapability*]

#### 8.3.8.1 Introduction

The detailed description of the [*dmCapability*] resource can be found in clause 5.8.12 of the oneM2M TS-0023 [3].

Table 8.3.8.1‑1: Data Type Definition of [*dmCapability*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmCapability,dmCapabilityAnnc | MAD-mod-dmCapability-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmCapability*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows enabling/disabling various capabilities on the Proximal IoT devices. The IPE shall create one [*dmCapability*] resource per capability.

#### 8.3.8.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmCapability ModuleClass, at least the mandatory ones.

If the Proximal IoT Technology allows enabling/disabling the capabilities of a Proximal IoT device, the IPE will create <*flexContainer*> [*enable*] and or [*disable*] specialization as child of the [*dmCapability*] resource.

#### 8.3.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.8.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.8.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.8.7 Resource [*enable*]

##### 8.3.8.7.1 Introduction

The detailed description of the [*enable*] resource can be found in clause 5.8.7 of the oneM2M TS-0023 [3].

Table 8.3.8.7.1‑1: Data Type Definition of [*enable*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| enable,enableAnnc | MAD-act-enable-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*enable*] resource is on discretion of the Originator IPE. It is created as child of a [*dmCapability*] resource.

##### 8.3.8.7.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.8.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.8.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.8.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.8.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall enable the capability on the Proximal IoT device.

Note: the steps of the enabling are left to the implementor IPE, but the IPE shall fill the *currentState* attribute of the parent [*dmCapability*] resource accordingly.

#### 8.3.8.8 Resource [*disable*]

##### 8.3.8.8.1 Introduction

The detailed description of the [*disable*] resource can be found in clause 5.8.7 of the oneM2M TS-0023 [3].

Table 8.3.8.8.1‑1: Data Type Definition of [*enable*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| disable,disableAnnc | MAD-act-disable-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*disable*] resource is on discretion of the Originator IPE. It is created as child of a [*dmCapability*] resource.

##### 8.3.8.8.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.8.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.8.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.8.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.8.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall disable the capability on the Proximal IoT device.

Note: the steps of the disabling are left to the implementor IPE, but the IPE shall fill the *currentState* attribute of the parent [*dmCapability*] resource accordingly.

### 8.3.9 Resource [*dmStorage*]

#### 8.3.9.1 Introduction

The detailed description of the [*dmStorage*] resource can be found in clause 5.8.13 of the oneM2M TS-0023 [3].

Table 8.3.9.1‑1: Data Type Definition of [*dmStorage*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| dmStorage,dmStorageAnnc | MAD-mod-dmStorage-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*dmStorage*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows handling various types of storage on the Proximal IoT devices. The IPE shall create one [*dmStorage*] resource per existing storage.

#### 8.3.9.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the dmStorage ModuleClass, at least the mandatory ones.

If the Proximal IoT Technology allows formatting/unmounting the storage(s) of a Proximal IoT device, the IPE will create <*flexContainer*> [*format*] and or [*unmount*] specialization as child of the [*dmStorage*] resource.

#### 8.3.9.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.9.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.9.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.9.6 Notification on update

No change from the generic procedures in clause 8.2.5.

#### 8.3.9.7 Resource [*format*]

##### 8.3.9.7.1 Introduction

The detailed description of the [*format*] resource can be found in clause 5.8.13 of the oneM2M TS-0023 [3].

Table 8.3.9.7.1‑1: Data Type Definition of [*format*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| format,formatAnnc | MAD-act-format-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*format*] resource is on discretion of the Originator IPE. It is created as child of a [*dmStorage*] resource.

##### 8.3.9.7.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.9.7.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.9.7.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.9.7.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.9.7.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall format the storage on the Proximal IoT device.

Note: the steps of the formating are left to the implementor IPE, but the IPE shall fill the attributes of the parent [*dmStorage*] resource accordingly.

#### 8.3.9.8 Resource [*unmount*]

##### 8.3.9.8.1 Introduction

The detailed description of the [*unmount*] resource can be found in clause 5.8.13 of the oneM2M TS-0023 [3].

Table 8.3.9.8.1‑1: Data Type Definition of [unmount]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| unmount,unmountAnnc | MAD-act-unmount-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*unmount*] resource is on discretion of the Originator IPE. It is created as child of a [*dmStorage*] resource.

##### 8.3.9.8.2 Create

No change from the generic procedures in clause 8.2.1.

##### 8.3.9.8.3 Retrieve

No change from the generic procedures in clause 8.2.2.

##### 8.3.9.8.4 Update

No change from the generic procedures in clause 8.2.3.

##### 8.3.9.8.5 Delete

No change from the generic procedure in clause 8.2.4.

##### 8.3.9.8.6 Notification on update

No change from the generic procedures in clause 8.2.5.

The IPE shall unmount the storage on the Proximal IoT device.

Note: the steps of the unmounting are left to the implementor IPE, but the IPE shall fill the attributes of the parent [*dmStorage*] resource accordingly.

### 8.3.10 Resource [*battery*]

#### 8.3.10.1 Introduction

The detailed description of the [*battery*] resource can be found in clause 5.3.1.10 of the oneM2M TS-0023 [3].

Table 8.3.10.1‑1: Data Type Definition of [*battery*]

|  |  |  |
| --- | --- | --- |
| Data Type ID | File Name | Note |
| battery,batteryAnnc | COD-mod-battery-vx\_y\_z.xsd |  |

NOTE: the creation and deletion of the [*battery*] resource is on discretion of the Originator IPE. It shall be created if the underlying Proximal IoT Technology allows monitoring the power supply on the Proximal IoT devices. The IPE shall create one [*battery*] resource per existing battery.

#### 8.3.10.2 Create

No change from the generic procedures in clause 8.2.1.

**Originator**: the IPE shall represent as many as possible datapoints of the battery ModuleClass.

#### 8.3.10.3 Retrieve

No change from the generic procedures in clause 8.2.2.

#### 8.3.10.4 Update

No change from the generic procedures in clause 8.2.3.

#### 8.3.10.5 Delete

No change from the generic procedure in clause 8.2.4.

#### 8.3.10.6 Notification on update

No change from the generic procedures in clause 8.2.5.

History

|  |
| --- |
| **Publication history** |
| V3.0.0 | April 2019 | Release 3 – Publication |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |