|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | RDM#53.2 |
| Source:\* | Cyrille Bareau, Orange, [cyrille.bareau@orange.com](mailto:cyrille.bareau@orange.com)  Bob Flynn, Exacta, [bob.flynn@exactagss.com](mailto:bob.flynn@exactagss.com)  Andreas Kraft, Deutsche Telekom, [a.kraft@telekom.de](mailto:a.kraft@telekom.de)  Marianne Mohali, Orange, [marianne.mohali@orange.com](mailto:marianne.mohali@orange.com) |
| Date:\* | 2022-02-16 |
| Reason for Change/s:\* | See the introduction. |
| CR against: Release\* | Release 4 |
| CR against: WI\* | Active WI-0109  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0033 v0.2.1 |
| Clauses \* | Modified clauses: 2.1, 3.2, 5, 6.2.2, 7.1, 7.3  New clause 8 |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Other TS/TR(s) impacted | N/A |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2020 (do not modify) | |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

If this is a correction, and the change applies to previous releases, a separate “mirror CR” should be posted at the same time as this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made e.g. a change impacting 5 tables should not only include a proposal to change only 3 tables. Include any changes to references, definitions, and abbreviations in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar.

Use change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the proposed new clause is located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to the content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

## Introduction

This draft is part of a series of CRs related to the Work Item WI-0109: IPE-based Device Management with FlexContainers. For a full introduction, see clause 2 “Justification” in WI-0109-IPE-based\_Device\_Management\_with\_FlexContainers-V0\_0\_1.DOCX.

In this specific draft, the proposed changes are as follows:

1. Add needed documents to the references list.
2. Added abbreviations.

3,4,5,6. Introduce new DM approach.

7 New clause to specify CRUD operations on SDT DM <flexContainers>.

### ----------------------- Start of change 1 -------------------------------------------

## 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".

### ----------------------- End of change 1 -------------------------------------------

### ----------------------- Start of change 2 -------------------------------------------

## 3.2 Abbreviations

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

DM Device Management

IoT Internet of Things

SDT Smart Device Template

### ----------------------- End of change 2 -------------------------------------------

### ----------------------- Start of change 3 -------------------------------------------

# 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.

### ----------------------- End of change 3 -------------------------------------------

### ----------------------- Start of change 4 -------------------------------------------

### 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.
* 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 for instance by creating under each such resource a <*subscription*> resource with *eventNotificationType* attribute set to “Blocking\_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

### ----------------------- End of change 4 -------------------------------------------

### ----------------------- Start of change 5 -------------------------------------------

## 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.

*NOTE: IPE-based DM with Smart Device Template (SDT), as defined in oneM2M TS-0001 [2] clause 6.2.4.1.2 and oneM2M TS-0023 [3] clause 5.8, is presented in this document. An alternate approach is to use CSE-based DM with <mgmtObj> as described in clause 10.2.8 of oneM2M TS-0001 [2] and Annex D of oneM2M TS-0004 [8].*

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 <flexContainer> specializations, mapping 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.

### ----------------------- End of change 5 -------------------------------------------

### ----------------------- Start of change 6 -------------------------------------------

## 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: 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] can be used. The approach described in this document is to use specialized <*flexContainer*> resource types as specified in oneM2M TS-0023 [3], based on the SDT data model. These resources shall be created by the responsible IPE as child resources of the <*node*> resource, which represents the managed device (see clause 7.1).
* Services defined in vertical domains specified in oneM2M TS-0023 [3] (agriculture, city, health, home, industry, railway, vehicular…): Specialized <*flexContainer*> resource types for moduleClasses 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 [2] 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.

### ----------------------- End of change 6 -------------------------------------------

### ----------------------- Start of change 7 -------------------------------------------

# 8 Device Management Operations

This clause specifies the procedures for managing device capabilities, using SDT DM <flexContainer> specializations.

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 <node> resource has a child [flexNode] specialization.

Throughout this clause, the expressions “Creator IPE” and “Managing IPE” refer to the IPEs that respectively creates and manages the NoDN. The Creator and Managing IPEs of the NoDN can be identical. If a DM resource has no associated Managing IPE, no DM operation could be performed on the corresponding NoDN. Creation and deletion rules could be applied, but are deployment specific and out of scope of this document. The association of a DM resource with a Managing IPE is out of scope of this document.

## 8.1 [*flexNode*] management

### 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 [2] 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 [3]. |
| Processing at Originator before sending Request | According to clause TS-0001 [2] 10.1.2.  The Originator should be a Creator IPE that manages the corresponding entity in the Proximal IoT System. |
| Processing at Receiver | According to clause TS-0001 [2] 10.1.2 |
| Information in Response message | All parameters defined in TS-0001 [2] table 8.1.3-1 apply with the specific details for:   * ***Content*:** Address of the created *[flexNode]* resource, according to clause TS-0001 [2] 10.1.2 |
| Processing at Originator after receiving Response | According to clause TS-0001 [2] 10.1.2 |
| Exceptions | According to clause TS-0001 [2] 10.1.2 |

#### 8.1.1.2 Create DM SDT modules

As specified in clause 5.8.2 of TS-0023 [3], 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.

For the generic operations on these DM SDT <flexContainer>, children of the [flexNode], see clause 8.2.

### 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 [2] table 8.1.2-3 apply with the specific details for:  ***Content*:** Void |
| Processing at Originator before sending Request | According to clause TS-0001 [2] 10.1.3 |
| Processing at Receiver | According to clause TS-0001 [2] 10.1.3 |
| Information in Response message | All parameters defined in TS-0001 [2] table 8.1.3-1 apply with the specific details for:  ***Content*:** Attributes of the *[flexNode]* resource as defined in clause TS-0023 [3] 5.8.2. |
| Processing at Originator after receiving Response | According to clause TS-0001 [2] 10.1.3 |
| Exceptions | According to clause TS-0001 [2] 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 [2] table 8.1.2-3 apply with the specific details for:  ***Content***: attributes of the *[flexNode]* resource as defined in clause TS-0023 [3] 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 [2] 10.1.4 |
| Processing at Receiver | According to clause TS-0001 [2] 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 [2] 10.1.4 |
| Processing at Originator after receiving Response | According to clause TS-0001 [2] 10.1.4 |
| Exceptions | According to clause TS-0001 [2] 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 [2] table 8.1.2-3 apply |
| Processing at Originator before sending Request | According to clause TS-0001 [2] 10.1.5  The Originator should be a Creator IPE that manages the corresponding entity in the Proximal IoT System. |
| Processing at Receiver | According to clause TS-0001 [2] 10.1.5 |
| Information in Response message | According to clause TS-0001 [2] 10.1.5 |
| Processing at Originator after receiving Response | According to clause TS-0001 [2] 10.1.5 |
| Exceptions | According to clause TS-0001 [2] 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 entity 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 [2] clause 10.1.2, the procedure in table 8.2.1-1 shall 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 [3] clause 5.8. |
| Processing at Originator before sending Request | The Originator should be a Creator 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. See note. |
| Processing at Receiver | According to clause 10.1.2 |
| 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 |
|  | |

Notes:

* the Creator IPE shall create at least the custom attributes that correspond to mandatory datapoints of SDT moduleClasses;
* it should create as many as possible custom attributes that correspond to optional datapoints of SDT moduleClasses;
* it should not create optional attributes that have no associated equivalent in the Proximal IoT Technology specific data model.

#### 8.2.1.1 Create <*subscription*>

Update requests to this created <*flexContainer*> shall be retargeted to the Managing IPE. For this, the IPE shall create a <*subscription*> resource, child of the <*flexContainer*> resource, with the attributes given in Table 8.2.1.1-1.

Table 8.2.1.1-1: <*subscription*> resource

| Attributes of *<subscription> resource* | Description / Value |
| --- | --- |
| *notificationURI* | IPE URI |
| *notificationContentType* | all attributes |
| *notificationEventType* | 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 [3] 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) |

Note: the Managing IPE is responsible for providing updated data to DM SDT <flexContainer> resources of the entities it manages (see clause 6.2.2).

### 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 [2] 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 [3] clause 5.8. |
| Processing at Originator before sending Request | None |
| Processing at Receiver | * Send a blocking UPDATE notification to the Managing IPE. * Wait for the notification response. * If the notification is successful, modify the current <flexContainer> with the attributes in the originating request. |
| Processing at Managing IPE | The Managing IPE is responsible for invoking the appropriate non-oneM2M Proximal IoT function(s). See NOTIFY procedures on each DM flexContainer specialization in clauses 8.3.\* |
| Information in Response message | The resource (or attributes defined in the originating request, depending on the result content) as it has been modified. |
| Processing at Originator after receiving Response | None |
| Exceptions | Error code if a timeout happens before the notification completes, otherwise returns the retargeted response.  Error code if the request is trying to modify a read-only attribute (see note).  Error code if the Managing IPE returned an error. |

NOTES:

* The creation and update of the <flexContainer>’s custom attributes that correspond to datapoints specified as R (read only) in clauses 5.8.x of TS-0023 [3] are allowed only to the Creator and Managing IPE.
* The <flexContainer>’s custom attributes that correspond to datapoints specified as RW are updatable by external AEs. This is ensured by the Managing IPE when it is notified about the 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 [2] 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 | The Originator should be a Creator or Managing IPE that manages the corresponding entity in the Proximal IoT Technology. |
| Processing at Receiver | According to clause 10.1.5 |
| 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 Notify on DM SDT *<flexContainer>*

Following the <*subscription*> resource defined in clause 8.2.1.1, the Managing IPE shall be notified when the<*flexContainer*>, 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. For specific behaviour, depending on the type of <flexContainer>, see clause 8.3.

If the triggered IoT function is not successful (timeout or returned error), the Managing IPE shall respond to the notification with an error, otherwise it will respond with success.

If the UPDATE request contains custom attributes that correspond to read-only datapoints, the Managing IPE shall raise an error.

If the UPDATE request contains custom attributes that correspond to optional read-write datapoints that are not present in the resource, the Managing IPE may raise an error (there can be exceptions to this rule, for instance the *description* attribute of the [dmDeviceInfo]).

## 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 Creator IPE.
* As specified in clause 5.8.2 of TS-0023 [3], 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 Creator IPE shall create 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 Notify

No change from the generic procedure in clause 8.2.5.

The only modifiable attributes are manufacturerDetailsLink, supportURL, presentationURL friendlyName and description.

Depending on the underlying Proximal IoT technology, the Managing may forward the request to the managed node using the appropriate IoT function.

### 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 Creator 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 Creator IPE shall create as many as possible datapoints of the dmAgent ModuleClass, at least the mandatory ‘state’ datapoint (attribute ‘*state*’), but it should not create optional attributes that have no associated equivalent in the Proximal IoT Technology specific data model.

As the ‘reboot’ SDT action is mandatory in clause TS-0023 [3] 5.8.3, the Creator 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 Creator 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 Notify

No change from the generic procedure in clause 8.2.5.

The only modifiable attribute is systemTime.

Depending on the underlying Proximal IoT technology, the Managing IPE may forward the request to the managed node using the appropriate IoT function.

#### 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger a reboot of the corresponding Proximal IoT device, according to the rebootType argument.
* 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

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, attempt 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 Creator 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 Creator 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 Creator IPE will create a <*flexContainer*> [*readIO*] specialization as child of the [*dmDataModelIO*] resource.

If the Proximal IoT Technology allows writing data model parameters, the Creator 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 Notify

No change from the generic procedure 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger a read of the parameter(s) referenced by the *address* argument on the corresponding Proximal IoT device.
* 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger a write of the parameter(s) referenced by the *address* argument, with values in the *payload* argument, on the Proximal IoT device.
* 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 Creator 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 Creator IPE shall create 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 Creator 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 Creator 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 Notify

No change from the generic procedure 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger an update of a new firmware, referenced by the *url* and *version* argument, on the Proximal IoT device.
* 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 Managing 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger a toggle between the current firmware and a backup one, on the Proximal IoT device.
* 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 Managing 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 |  |

NOTE 1: the creation and deletion of the [dmSoftware] resource is on discretion of the Creator 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.

NOTE 2: 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 Creator IPE shall create 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 Creator 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 Notify

No change from the generic procedure 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger an activation of the software on the Proximal IoT device.

Note: the steps of the activation are left to the Managing 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger a deactivation of the software on the Proximal IoT device.

Note: the steps of the deactivation are left to the Managing 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 |  |

NOTE 1: the creation and deletion of the [dmPackage] resource is on discretion of the Creator 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 Creator IPE shall create one [dmPackage] resource per package.

NOTE 2: 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 Creator IPE shall create 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 [3] 5.8.9, the Creator 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 Notify

No change from the generic procedure 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger an installation of the package on the Proximal IoT device.

Note: the steps of the installation (download, install, etc.) are left to the Managing 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 Creator 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.

##### 8.3.6.7.6 Notify

Processing at Managing IPE:

* Trigger a deinstallation of the package on the Proximal IoT device.

Note: the steps of the deinstallation are left to the Managing 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 [*update*]

|  |  |  |
| --- | --- | --- |
| 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger an update of the package on the Proximal IoT device.

Note: the steps of the update (download, install, etc.) are left to the Managing 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 Creator 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 Creator IPE shall create 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 Creator 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 Notify

Processing at Managing IPE:

* When the *enabled* attribute is set to TRUE (resp. FALSE), trigger a start (resp. a stop) of the logging behaviour on the Proximal IoT device.
* Modify the *enabled* attribute if the triggered action succeeds.
* 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 Creator 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 Creator IPE shall create 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 Notify

Processing at Managing IPE:

* Trigger a retrieval of the logs of the Proximal IoT device.

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

When retrieval is completed, the Managing 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 Creator IPE. It shall be created if the underlying Proximal IoT Technology allows enabling/disabling various capabilities on the Proximal IoT devices. The Creator 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 Creator IPE shall create 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 Creator 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 Notify

No change from the generic procedure 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger the enabling of the capability on the Proximal IoT device.

Note: the steps of the enabling are left to the Managing 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger the disabling of the capability on the Proximal IoT device.

Note: the steps of the disabling are left to the Managing 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 Creator 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 Creator IPE shall create 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 Creator 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 Notify

No change from the generic procedure 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger the formatting of the storage on the Proximal IoT device.

Note: the steps of the formating are left to the Managing 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 Creator 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 Notify

Processing at Managing IPE:

* Trigger the unmount of the storage on the Proximal IoT device.

Note: the steps of the unmounting are left to the Managing 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 Creator 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 Creator IPE shall create 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 Notify

No change from the generic procedure in clause 8.2.5.

### ----------------------- End of change 7 -------------------------------------------