|  |  |
| --- | --- |
| Input Contribution | |
| Meeting ID\* | SDS#45 |
| Title:\* | TR-0064 |
| Source:\* | JaeSeung Song, KETI, [jssong@sejong.ac.kr](mailto:jssong@sejong.ac.kr)  Sherzod Elamanov, KETI, [selamanov@gmail.com](mailto:selamanov@gmail.com) |
| Date:\* | 2020-04-23 |
| Input related to\* | TR-0064 ZigBee Interworking |
| Intended purpose of  document:\* | Decision  Discussion  Information  Other <specify> |
| Impacted other TS/TR(s) | N/A |
| Decision requested or recommendation:\* | Add Section 7 ZigBee interworking procedures |
| Template Version: January 2017 (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.

# Introduction

This contribution proposes to add a new section for ZigBee Interworking procedures.

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

# 7 Possible Solutions for oneM2M and ZigBee Interworking

### 7.1 Summary of Interworking Architecture for exposure of ZigBee Functions

The interworking architecture between oneM2M and ZigBee is established through ZigBee IPE. The ZigBee IPE serves as a gateway between oneM2M and ZigBee. It consists of oneM2M AE and ZigBee Coordinator. The ZigBee IPE that exposes ZigBEe Functions to the oneM2M System is responsible for originating a Zigbee network, registering new devices, creation of oneM2M resources representing the exposed ZigBee functions on its Registrar CSE, translating requests between oneM2M and ZigBee. A single ZigBee IPE may support one or more ZigBee devices and expose them to the oneM2M System.

ZigBee uses ZigBee Cluster Library (ZCL) to represent functionalities of devices through a set of attributes or commands. In the oneM2M System, TS-0023 describes a templating tool for describing heterogenous devices and their functionalities Smart Device Template (SDT). SDT offers a generic and flexible modeling structure for non-oneM2M devices including ZigBee devices. Therefore, a mapping between ZCL and SDT is applied in this architecture and mapping rules will be described..

### 7.2 Registration

#### 7.*2*.1 <AE> resource representing a *ZigBee* IPE

The first step to support the Zigbee interworking with the oneM2M System is to register a ZigBee IPE to its Registrar CSE as an <*AE*> resource. As soon as Zigbee Coordinator establishes and runs the network, it shall register at CSE by initiating an <*AE*> Create request, an <*AE*> resource representing that ZigBee IPE is created as a result of the registration. This resource is a parent for <*flexContainer*> resource specializations representing ZigBee devices paired to an associated ZigBee Coordinator. These devices are modelled using an SDT Device (details are described in the next section). Figure 7.2.2.1-1 shows an example resource tree structure of the *ZigbBee IPE* <*AE*> 2resource. *ZigBee IPE* has a [*deviceTemperatureSensror*] as a child resource, which represents a Zigbee device with temperature and humidity sensors.

#### 7.*2*.2 deviceTemperatureSensror as a <flexContainer> resource specialization representing a ZigBee Device

ZigBee devices shall be modelled as SDT Devices. Mapping of the SDT Device model to oneM2M resources is performed according to the general mapping procedure described in clause 6.2.2 of TS-0023. An SDT Device component is mapped to a specialization of a <*flexContainer*> resource with an associated 'DeviceClass ID' (e.g. "org.onem2m.home.device.tv") *containerDefinition* attribute.

Figure 7.2.2-1 shows an example of a ZigBee device: *[deviceTemperatureSensror],* which is modelled as *a <flexContainer>* resource specialization derived from the corresponding SDT Device component. The model of *[deviceTemperatureSensror]* follows the schema described in clause 5.5.45 of TS-0023.

ZigBee IPE

[deviceTemperatureSensor]

<subscriptions>

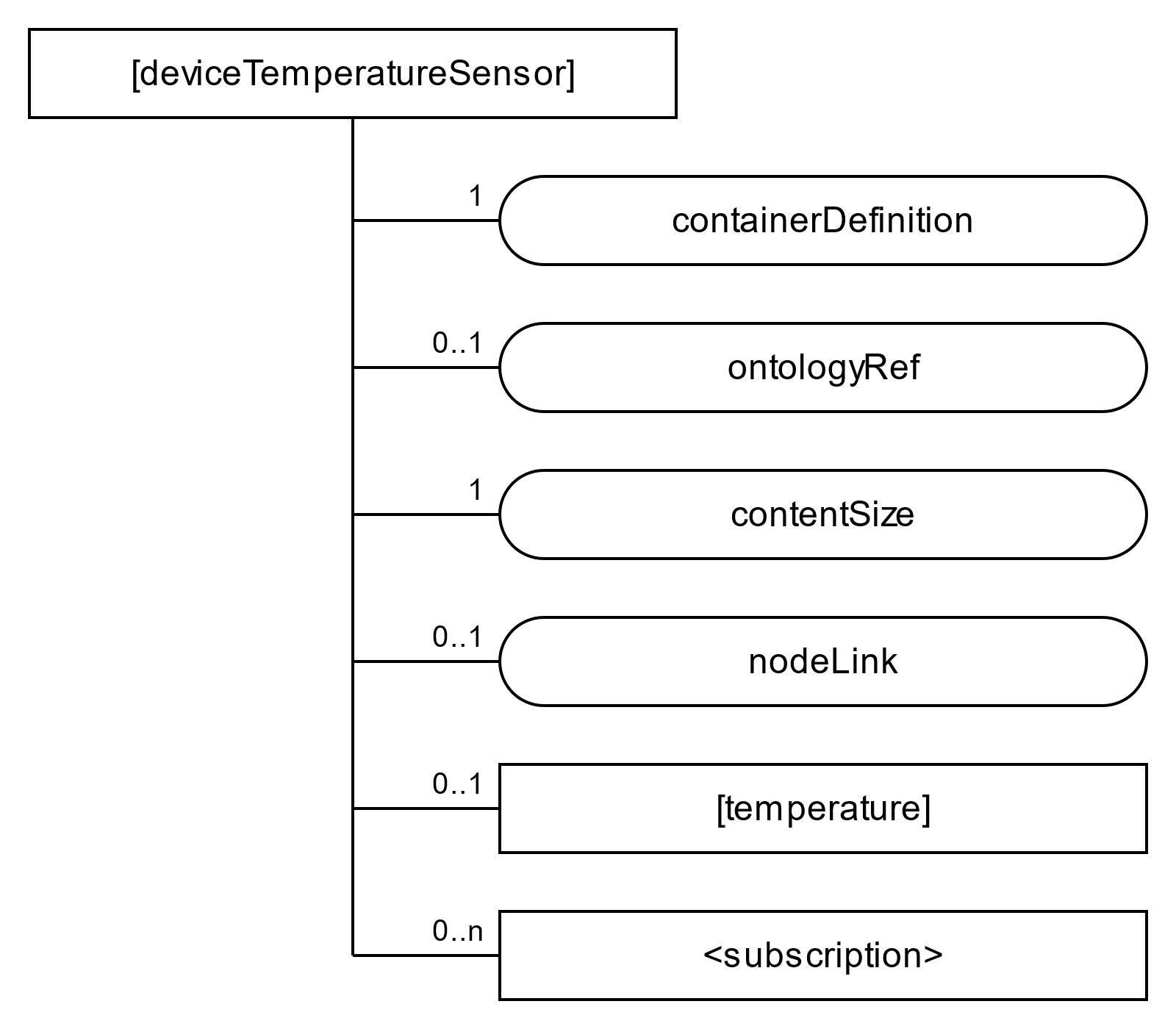
0..n

0..n

**Figure 7.2.1-1 *<AE>* resource representing a ZigBee IPE**

7.2.2 deviceTemperatureSensor as a <flexContainer> resource specialization representing a ZigBee Device

ZigBee devices shall be modelled as SDT Devices. Mapping of the SDT Device model to oneM2M resources is performed according to the general mapping procedure described in clause 6.2.2 of TS-0023. An SDT Device component is mapped to a specialization of a <*flexContainer*> resource with an associated 'DeviceClass ID' (e.g. "org.onem2m.home.device.tv") *containerDefinition* attribute.

Figure 7.2.2-1 shows an example of a ZigBee device:*[* *deviceTemperatureSensor],* which is modelled as *a <flexContainer>* resource specialization derived from the corresponding SDT Device component. The model of *[deviceTemperatureSensor]* follows the schema described in clause 5.5.45 of TS-0023.

**Figure 7.1.2-1 *[deviceTemperatureSensor]* resource representing a ZigBee Slave**

7.2.3 Deriving SDT ModuleClass and DataPoints and from ZCL Clusters and Attributes.

In order to enable interworking, a mapping between ZCL and SDT is defined. ZCL Clusters are similar to SDT ModuleClasses. A ZigBee device can have multiple Clusters depending on its functionality. This feature matches the property of SDT where a Device component can have multiple Modules. Furthermore, ZCL Clusters have Attributes similarly as SDT ModuleClasses have DataPoints. For example, *Temperature Measurement* Cluster (Cluster ID: 0x0402) is mapped to Temperature ModuleClass, and corresponding Attributes (e.g. measuredTemperature) are mapped to DataPoints. However, not all ZCL Clusters can be converted to SDT ModuleClasses. For example, Cluster *Basic* (Cluster ID: 0x0001) contains information like device model, manufacture name, hardware version etc. This cluster shall be mapped to [*deviceInfo*] resource as described in Rule 1-3 of clause 6.2.2 in TS-0023. In case.ZCL Cluster does not logically match any ModuleClass provided in TS-0023, a custom ModuleClass shall be created. Some example mappings of prominent Clusters to ModuleClasses are provided in Table 7.2.3.1

**Table 7.2.3-1 Mapping between ZCL Clusters and SDT ModuleClasses**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cluster** | **Cluster ID** | **ModuleClass** | **Description** |
| Power Configuration | 0x0001 | Battery /  Mains (custom) | Device’s power source(s) to be determined and under/ over-voltage alarms to be configured |
| On/Off | 0x0006 | BinarySwitch | Allows a device to be put into the ‘on’ and ‘off’ states, or toggled between the two states |
| Level Control | 0x0008 | AudioLevel /  Brightness /  OpenLevel | Allows control of the level of a physical quantity |
| Temperature Measurement | 0x0402 | Temperature | Provides an interface to a temperature measuring device, allowing the configuration of measuring and the reporting of measurements. |
| Relative Humidity Measurement | 0x0405 | RelativeHumidity | Provides an interface to a humidity measuring device, allowing the configuration of relative humidity measuring and the reporting of measurements. |
| Colour Control | 0x0300 | Colour | Can be used to adjust the colour of a light |

The Temperature measurement Cluster has 4 attributes*: MeasuredValue, MinMeasuredValue, MaxMeasuredValue, Tolerance*. The *Tolerance* attribute does not have analogue DataPoint in TS-0023, so the ModuleClass can be extended by adding this attribute. These attributes are mapped to SDT DataPoints as following:

**Table 7.2.3-2 Mapping between ZCL Attributes and SDT DataPoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Attribute ID** | **Attribute Type** | **DataPoint** | **DataPoint Type** |
| MeasuredValue | 0x0000 | Int16 | currentTemperature | xs:float |
| MinMeasuredValue | 0x0001 | Int16 | minValue | xs:float |
| MaxMeasuredValue | 0x0002 | Int16 | maxValue | xs:float |
| Tolerance | 0x0003 | Int16 | tolerance  (custom) | xs:float |

7.2.4 Temperature as a <flexContainer> resource specialization representing an SDT Module for a ZigBee device

Depending on the functionalities of a ZigBee device, one or more ModuleClasses, which are generic ModuleClasses, defined in TS-0023 can be used to design a ModuleClass for the target ZigBee device. The derived ModuleClass describes all functional capabilities of the target Mobus device.

For the [*deviceTemperatureSensor*] example described in clause 7.1.2.2 the Temperature ModuleClass (see clause 5.3.76 in TS-0023) is used. DataPoints of a parent ModuleClass (in this example Thermometer) are created according to the mapping rule described in clause 7.1.2.3.

ModuleClass is mapped into <*flexContainer*> resource specialization, for example Temperature, and its data points are mapped into *customAttributes* of that <*flexContainer*> resource specialization. However, those ModuleClasses do not consider interworking options with a non-oneM2M Device Nodes (noDN) such as ZigBee devices. For that reason, *nodnProperties* is added as a *customAttribute* of a <*flexContainer*> resource which is mapped from an associated ModuleClass model.

The *nodnProperties* attribute stores one-to-one mappings in serialized string format (e.g. JSON) between each DataPoint and a ZCL attribute from which it is created. Particularly, *nodnProperties* contains *clusterID, attributeID* attributes for each DataPoint. An example content of *nodnProperties* is shown on Figure **7.2.4-1.**

{"currentTemperature": {

"clusterID": "0x0402",

"attributeID": "0x0000"

},

"minValue": {

"clusterID": "0x0402",

"attributeID": 0x0001"

},  
"maxValue": {

"clusterID": "0x0402",

"attributeID": 0x0002"

},

"tolerance": {

"clusterID": "0x0402",

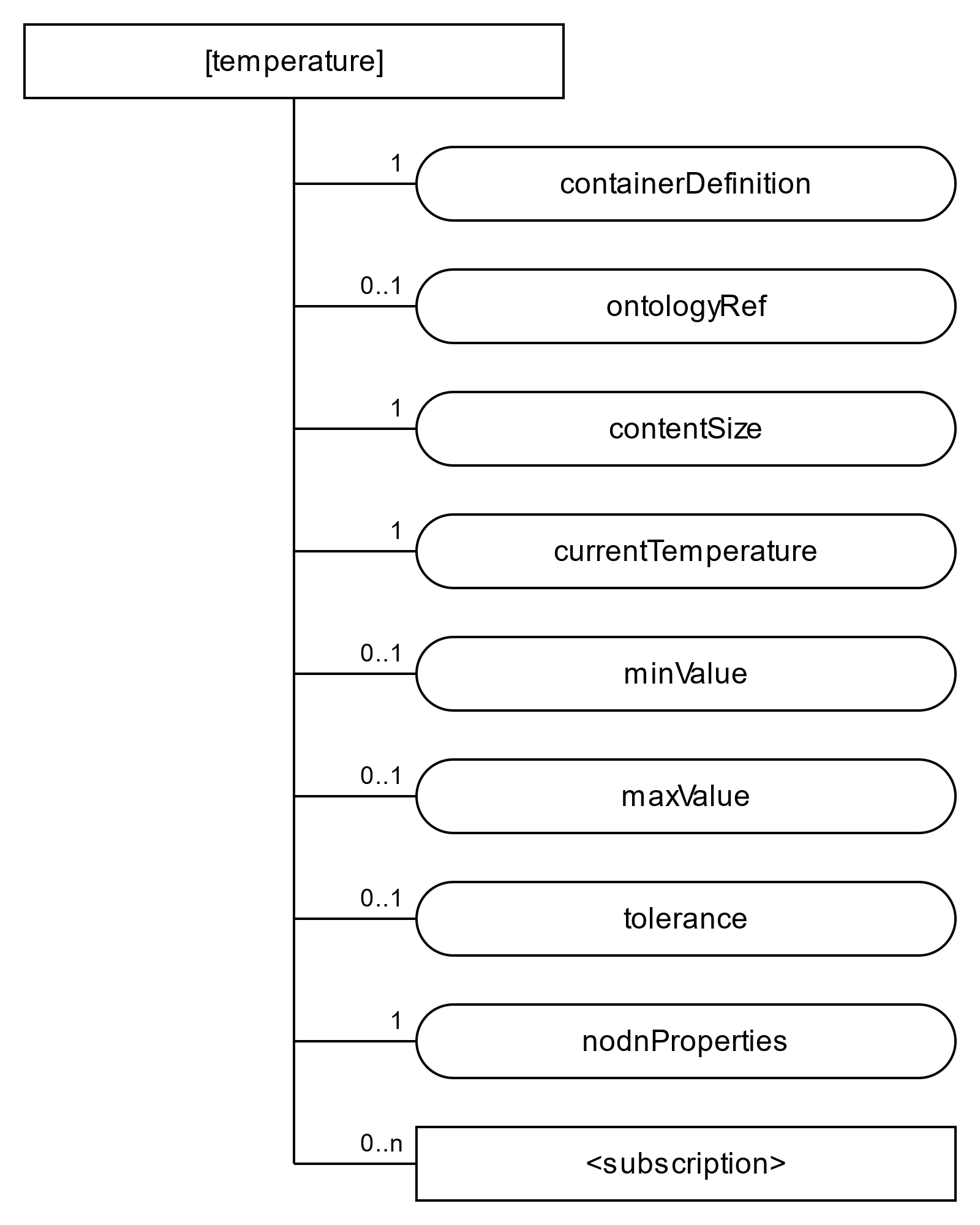
"attributeID": 0x0003"

}

}

**Figure 7.2.4-1 An example contents of *noDNproperties***

**F**igure 7.2.4-2 shows an example of a Module, the specialization for TemperatureModuleClass*,* which isa *<flexContainer>* resource specialization derived from a corresponding SDT Module component. Here the DataPoints of the *[Temperature]* resource specialization are derived as a result of the mapping procedures described in clause 7.1.4. *nodnProperties* [*customAttribute*] is added to support ZigBee interworking.

**Figure 7.2.4-2** ***[Temperature]* as a <*flexContainer*> resource specialization representing an SDT Module for ZigBee device**

### 7.2 Device registration

7.2.1 ZigBee device registration call flow

Figure 7.2.1-1 shows the device registration call flow

1. ZigBee IPE sends a Create *<AE>* request to a Hosting CSE to register the ZigBee-IPE (see clause 7.4.5.2.1 in TS-0004). The request must specify *App-ID* and *requestReachabily* attributes of the to be created <*AE*> resource. Other <*AE*> attributes are optional.
2. After verifying the privileges and the given attributes, the Hosting CSE creates the <*AE*> resource.
3. Hosting CSE responds with the successful result of *<AE>* resource creation, otherwise it responds with an error.
4. The device attempts to pair to Zigbee Coordinator by factory resetting.
5. The ZigBee IPE runs pairing interview process. On successful pairing, the Zigbee IPE sends a pairing response.
6. The device broadcasts Device announcement request. It announces its n*etwork address*, *extended address*, and *capability information* (e.g. full-function device, AC power etc.) to all devices in the network.
7. The ZigBee IPE sends Node Descriptor request (e.g. device type, band frequency etc.)
8. The device responds for Node Descriptor request
9. The ZigBee IPE requests active endpoints list on the device
10. The device responds for active endpoints request
11. For each endpoint on the device, the ZigBee IPE sends a Simple Descriptor request to collect information on profiles and clusters.
12. The device responds for the Simple Descriptor request
13. Depending on application needs, the ZigBee IPE requests other information which is required for interworking (e.g. device model, manufacturer name etc). Read Attributes request is performed for this purpose.
14. The device responds for the Read Attributes request.
15. The ZigBee IPE sends a Create <*flexContainer*> resource for ZigBee device to CSE as described in clause 7.1.2.
16. The CSE creates a <*flexContainer*> resource.
17. The CSE responds to the IPE if the creation was successful.
18. The ZigBee IPE sends requests to a Hosting CSE to create <*flexContainer*> resource for each SDT Module as described in section 7.1.4. For all <*flexContainer*> resources, the *containerDefinition* attribute is mandatory*.* The *contentSize* attribute is calculated by Hosting CSE. *CustomAttributes* must be specified if they are mandatory for that <*flexContainer*>.

For the presented above temperature sensorexample, *[deviceTemperatureSensor], [temperature]* as child resources of *[deviceTemperatureSensor],* and <*subscription*> resources shall be created.

1. After verifying the privileges and the given attributes, the Hosting CSE creates each resource.
2. Hosting CSE responds with the successful result for each created resource, otherwise it responds with an error.

****

**Figure 7.2.1-1 Device registration call flow**

### 7.3 Retrieve data from a ZigBee end device

Suppose a scenario when current readings of a ZigBee end device need to be displayed at an AE application. The device continuously reports data to the ZigBee IPE, then the IPE uploads that data to a CSE hosted on a server in the network. Initially, AE shall be subscribed to the <*flexContainer*> resource, which is a specialization of some SDT module for a Zigbe device (e.g. Temperature, see clause 7.1.2.4), using a <*subscription*> resource (*notificationEventType A,* see clause 9.6.8 in TS-0001). The following steps described in the Figure 7.3-1 shall be performed for this scenario:

1. ZigBee IPE sends a retrieve <*flexContainer*> request to a hosting CSE. This <*flexContainer*> resource is a specialization of some SDT Module and contains *nodnProperties* attribute.
2. Hosting CSE responds to the retrieve request with <*flexContainer*> data that includes *nodnProperties*.
3. ZigBee device repeatedly sends Report attributes request (ZCL frame, command: 0x0a).
4. The ZigBee IPE receives the messages from the device and uses information in *nodnProperties* to map the ZigBee messages into oneM2M messages. The ZigBee IPE sends an update <*flexContainer*> request (see clause 7.4.37.2.3 in TS-0004). The request body specifies the *customAttributes* to be updated and their new values read from the ZigBee device.
5. After verifying the privileges and the given attributes, the hosting CSE updates <*flexContainer*> resource.
6. The hosting CSE responds with updated <*flexContainer*> data after successful update to the ZigBee IPE, otherwise it responds with an error.
7. The hosting CSE sends a notification for <*flexContainer*> resource update to the AE (see clause 7.5.1.2.2 in TS-0004).
8. The AE sends a confirmation message about notification receiving to the hosting CSE (see clause 7.5.1.2.2 in TS-0004).



**Figure 7.3-1 ZigBee End Device monitoring call flow**

### 7.4 Write data to a ZigBee device

Suppose a scenario when it is required to update some value in a ZigBee device through an AE application registered to a CSE. Initially, the ZigBee IPE shall be subscribed to the <*flexContainer*> resource, which is a specialization of some SDT module for a ZigBee device (e.g. Temperature, see clause 7.2.4), using a blocking type of <*subscription*> resource (*notificationEventType G,* see clause 9.6.8 in TS-0001). The following steps described in the Figure 7.4-1 shall be performed for this scenario:

1. In order to write data to a ZigBee device from an AE, the AE sends a request to update specified *customAttributes* of the *<flexContainer>* resource which map to the ZCL attribute (see clause 7.4.37.2.3 in TS-0004).
2. After verifying the privileges and the given attributes, the hosting CSE sends a notification for the received write request to the ZigBee IPE (notification includes *nodnProperties*) and temporarily blocks the *<flexContainer>* resource for any UPDATE operations (see clause 7.5.1.2.2 in TS-0004).
3. ZigBee IPE uses information stored in *nodnProperties* to identify *network address, destination endpoint, cluster, profile, and command*. Afterwards, the IPE sends a ZCL message to the device. ()
4. ZigBee device responds with written data to ZigBee IPE.
5. ZigBee IPE responds to the hosting CSE with successful device update message, otherwise responds with an error (see clause 7.5.1.2.2 in TS-0004).
6. If the device was updated successfully, the hosting CSE updates the *<flexContainer>* resource internally, otherwise discards the changes. The resource is unlocked for UPDATE operations.

The hosting CSE responds to AE with the result of UPDATE request



**Figure 7.4-1 Writing to a ZigBee Device call flow**

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