TTC仕様書
Technical Specification

TS-M2M-0030v3.0.2

oneM2M 技術仕様書
オントロジーベースのインターワーク

oneM2M Technical Specification
Ontology Based Interworking

2019 年 06 月 28 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TS-M2M-0030v3.0.2

oneM2M 技術仕様書－オントロジーベースのインターワーク [oneM2M Technical Specification - Ontology Based Interworking]

＜参考＞ [Remarks]

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

適用レベル [Application level]：E2

本標準の本文、付属資料および付録の文章および図に英文記述を含んでいる。

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

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

本標準は、oneM2M で承認された Technical Specification 0030V3.0.2 に準拠している。

[This standard is standardized based on the Technical Specification 0030 (V3.0.2) approved by oneM2M.]

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

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

原標準が参照する標準のうち、TTC 標準に置き換える項目。

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

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

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

４．工業所有権 [IPR]

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

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

５．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

-

ONEM2M
TECHNICAL SPECIFICATION

Document Number	TS-0030-V-3.0.2
Document Name:	Ontology based Interworking
Date:	2019-04-18
Abstract:	The present document specifies Generinc Interworking of the oneM2M System with external systems (e.g. Area Networks containing non-oneM2M devices) that can be described with ontologies that are compliant with oneM2M's Base Ontology in TS-0012.

Template Version: 08 September 2015 (Do not modify)

-

About oneM2M

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

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

-

Contents

-

-

1 Scope

The present document specifies Generinc Interworking of the oneM2M System with external systems (e.g. Area Networks containing non-oneM2M devices) that can be described with ontologies that are compliant with oneM2M's Base Ontology, specified in oneM2M TS-0012 [3].

In oneM2M Release 2 the specification for Ontology based Interworking had been contained in clauses 8 and 9 of oneM2M TS-0012-v2.2.0 [4].

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-0012: "Base Ontology".

[4] oneM2M TS-0012-v2.2.0: "Base Ontology".

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

[6] oneM2M TS-0014: "LWM2M Interworking".

[7] oneM2M TS-0024: "OIC Interworking".

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

2.2 Informative references

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

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

[i.1] oneM2M Drafting Rules.

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

[i.2] Smart Appliances REFerence (SAREF) ontology.

NOTE: Available at http://ontology.tno.nl/saref.

-

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-0012 [3] and the following apply:

Abstract Device: virtual Device (i.e. a set of oneM2M resources together with an IPE) that allows a communicating entity to communicate with an Interworked Device, using an Abstract Information Model, without the need to know the technology specific Device Information Model of that Interworked Device

Abstract Information Model: Information Model of common functionalities abstracted from a set of Device Information Models (see [1])

Abstraction: process of mapping between a set of Device Information Models and an Abstract Information Model according to a specified set of rules (see [1])

Abstraction Application Entity: specialized AE that communicates with an IPE and facilitates Abstraction by providing Services that translate between the Abstract Information Model and the Device Information Model of the IPE

Communicating Entity: oneM2M entity (usually an AE) that communicates with the IPE for the purpose of sending/receiving data from the Interworked Device

Device Information Model: Information Model of the native protocol (e.g. ZigBee) for the physical device (see [1])

Interworked Device: non-oneM2M device (NoDN) for which communication with oneM2M entities can be achieved via an Interworking Proxy Application Entity (IPE) (see [3])

Interworking Proxy Application Entity: specialized AE that facititates interworking between Non-oneM2M Nodes (NoDN) and the oneM2M System.

NOTE: An IPE maps data of the NoDN into oneM2M resources (Interworked Devices). It invokes operations in the NoDN when the related oneM2M resources are modified and modifies oneM2M resources based on the output of NoDN operations (see [1])

Proxied Device: virtual Device (i.e. a set of oneM2M resources together with an IPE) that represents the Interworked Device in the oneM2M System (see [3])

Ontology based Interworking: Ontology based Interworking allows interworking with many types of non- oneM2M Area Networks and Devices that are described in the form of a oneM2M compliant ontology which is derived from the oneM2M Base Ontology (see [3])

NOTE: Ontology based Interworking supports the interworking variant "full mapping of the semantic of the non-oneM2M data model to Mca" as indicated in clause F.2 of oneM2M TS-0001 [2].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in oneM2M TS-0011 [1], oneM2M TS-0012 [3] and the following apply:

CE	Communicating Entity
IPE	Interworking Proxy Application Entity (see [1])

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 to Ontology based Interworking (informative)

5.1 Basic concepts of Ontology based Interworking

5.1.1 Ontology based Interworking vs. Specific interworking

oneM2M supports interworking with several specific non-oneM2M solutions. Examples are: LWM2M Interworking (oneM2M TS-0014 [6]) or OIC Interworking (oneM2M TS-0024 [7]). While these examples refer to specific technologies oneM2M also allows to specify only data models - e.g. in oneM2M TS-0023: "Home Appliances Information Model and Mapping" [5] - which does not assume that a specific technology is used. The data model in oneM2M TS-0023 could e.g. be implemented with 'native' oneM2M entities like ASNs, ADNs and MNs or it could just as well be implemented in a non-oneM2M solution that is interworked with oneM2M via an Interworking Proxy Application Entity (IPE).

Ontology based Interworking is taking an approach similar to oneM2M TS-0023, however in this case the data model is not specified but can flexibly be provided in form of an ontology. That ontology needs to be formally described (e.g. in OWL format).

Ontology based Interworking can be used in cases where oneM2M does not provide a standardized datamodel but still interworking is desired. Such a situation may arise if e.g. a company wants to publish their proprietary datamodel for interworking purposes but does not wish to reveal their proprietary technology (radio technology, communication protocol) for data transmission.

For Ontology based Interworking the ontology that describes the data model of the interworked technology needs to be provided in the oneM2M solution. This ontology enables the IPE to create specific resourcetypes (specializations of <*flexContainer*>), through dynamically created XSDs that are derived from the ontology.
From these resourcetypes oneM2M resources are created by the IPE for communication of oneM2M communicating entities with the IPE.

As with any other form of interworking, the IPE provides the translation of data in these resources into/from the external technology.

5.1.2 Use of ontologies for Ontology based Interworking with Area Networks

Interworking with Area Networks is accomplished in oneM2M through functionality provided by Interworking Proxy Entities (IPE).

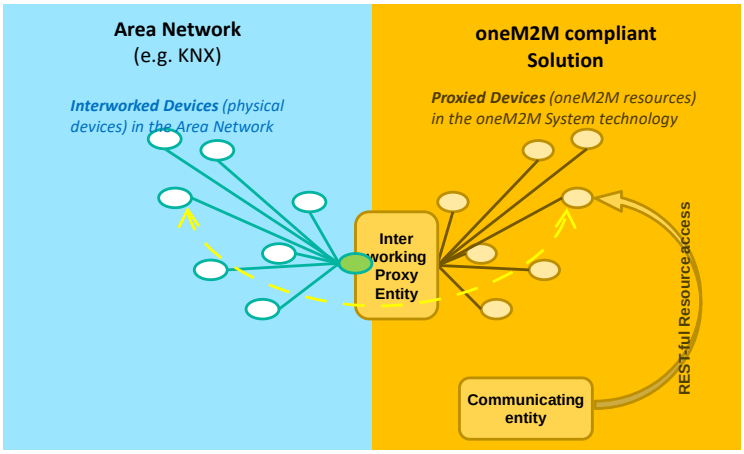

Figure 1: Interworking

-

The IPE creates "proxied" devices as oneM2M Resources (e.g. AEs) in the oneM2M Solution that can be accessed by communicating entities (e.g. oneM2M Applications) in the usual way.

To accomplish the creation of proxied devices the IPE uses an ontology that describes the Device Information Model of the interworked Area Network and its entities (device types, their operations, etc.).
For example, in figure 1, an ontology that describes a KNX Area Network and its entities would be needed.

To achieve the flexibility for the IPE to create proxied devices for many different types of Area Networks each ontology that describes a specific Device Information Model needs to be derived from the Base Ontology that is specified in [3]. This allows to specify a common scheme of mapping the classes of the ontology that describes the Device Information Model into oneM2M resources (see clause 7).
E.g. the OWL representation of an ontology that describes the Device Information Model of an Area Network of type "KNX" needs to:

a) contain an 'include' statement which includes Base Ontology;

b) the Class of "KNX Nodes" needs to be a subclass of the "Device" Class of oneM2M's Base Ontology;

c) the Class of "KNX Communication Objects" needs to be a subclass of the "Service" Class of the Base Ontology;

d) etc.

NOTE: For the purpose of Ontology based Interworking with Area Networks ***the Base Ontology is only used to describe type information and not for describing instances*** of these types. E.g. the Base Ontology describes the type "Device", but does not contain information about a specific device.The Base Ontology therefore only contains Classes and Properties but not instances.
That principle needs to be followed by the ontology that describes the Device Information Model.

5.2 Using Ontology based Interworking with Device Abstraction

5.2.1 General description

As explained in clause 5.1 it is the task of an IPE to interact via the Area Network with the Interworked Devices and to provide oneM2M resources (Proxied Devices) to the communicating entities for communication with the Interworked Devices. However these Proxied Devices still exhibit the native data model - the Device Information Model of the external technology of the device - and a communicating entity needs to know that native Device Information Model (e.g. ZigBee, KNX… information model).

Device abstraction relieves a communicating entity that wants to communicate with an Interworked Device (e.g. a ZigBee device) from the need to know the native Device Information Model of that Interworked Device.

Additionally to providing interworking, the IPE may translate between the - technology specific - native Device Information Model and an Abstract Information Model, that is based on of common functionalities abstracted from a set of Device Information Models. Such Abstract Information Models can be provided by industry associations of a specific industry sector.
An example of an Abstract Information Model, which is specified in oneM2M is the Home Appliance Information Model (HAIM), specified in oneM2M TS-0023 [5].

As in the case of a native Device Information Model that is used by the IPE also an Abstract Information Model can be described by an ontology and that ontology needs to be derived from the Base Ontology.

5.2.2 An example, involving ZigBee, HAIM and SAREF

Figure 2 illustrates this situation for a light switch. In the example the physical implementation is a ZigBee device implementing a ZigBee Service "On/Off Cluster". An IPE for ZigBee creates the interworking towards the ZigBee network.

To enable abstraction, this device is abstracted as oneM2M device according to the Home Appliance Information Model (HAIM). In HAIM the corresponding Service is a "binary Switch".

Both types of Services expose a Function "On Off Function" which is e.g. described in the SAREF ontology.

-

To turn the switch on SAREF defines an "On Command".

The corresponding Service in HAIM is executed by setting an Input Datapoint called "powerState" to the binary value "TRUE".

In Zigbee an operation (ZigBee command) needs to be invoked in the On/Off Cluster with an input parameter (ZigBee Command ID) equal to 0.

A VariableConversion can been specified in the ontology of the ZigBee Device Information Model that contains the rules how to convert a value of InputDataPoint "powerState" into a value of OperationInput "ZigBee Command ID".

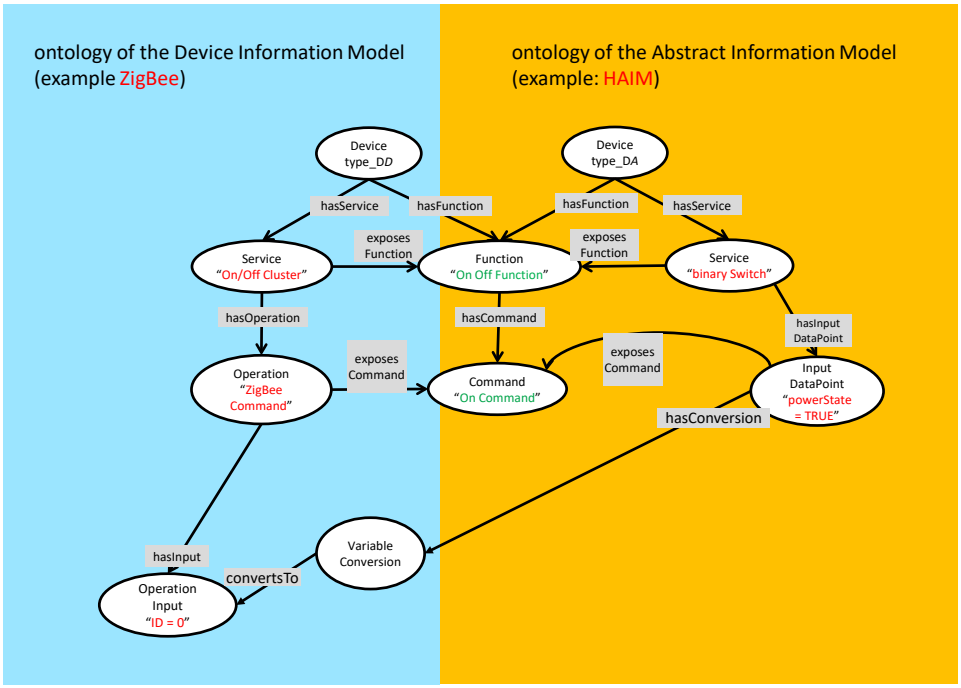

Figure 2: Ontologies relations

5.3 Priciples of data flows

5.3.1 Preconditions on the communicating entity

Data flows between a communicating entity and the IPE involve oneM2M resources - specialized <*flexContainer*>s and possibly <*AE*>s - that were created by the IPE for the purpose of that communication.

NOTE 1: In this clause and in the subsequent clauses it is assumed that the communicating entity and the IPE uses the oneM2M subscribe/notify mechanism to become informed about UPDATEd resources.
It remains, however, an implementation option if subscribe/notify is used or other mechanisms (polling or other mechanisms, e.g. in case of IPE collocated/integrated in its hosting CSE) are used.

1) Any communicating entity, that:

   a. wants to discover interworked non-oneM2M devices of the non-oneM2M system via the IPE needs to be subscribed to the <*AE*> resource of the IPE to get notified about newly created resources for Proxied Devices.
   These resources are created by the IPE to represent interworked non-oneM2M devices that were discovered by the IPE.

      - A communicating entity also needs to be subscribed to the <*AE*> resource of the IPE if it wants to use network services (e.g. broadcast services, registration services …) that are offered by the IPE.

-

NOTE 2: Since the IPE, in addition to being a oneM2M *AE*, is part of the non-oneM2M system it contains a non-oneM2M device that can offer network services.

b. wants to communicate with a specific interworked non-oneM2M device via the IPE needs to be subscribed to the <*flexContainer*> (or <*AE*>) resource that had been created by the IPE to represent that interworked non-oneM2M device as Proxied Device.

c. wants to communicate with a sub-device of a Proxied Device, represented by a <*flexContainer*> that is a child-resource of the Device's <*flexContainer*> (or <*AE*>) resource the communicating entity needs also to be subscribed to that <*flexContainer*> child-resource.

2) The communicating entity needs also be subscribed to:

a. The <*flexContainer*> resources, representing Services, that have been created by the IPE as child resourses of the resource of the Proxied Device. The attribute *notificationContentType* of the <*subscription*> needs to be set to "modified-attributes".

b. The <*flexContainer*> resources for the Operation invocation. These resources are child-resources of their respective Service <*flexContainer*> resources.
The *eventNotificationCriteria* conditions the *notificationEventType* needs to contain:
- (B) Deletion of the subscribed-to resource
  … to get notified when theoperation becomes unavailable

and, if the operation involves an answer (operation results). also
- (C) Creation of a direct child of the subscribed-to resource
  … to get notified about the creation of a <*flexContainer*> for the Operation result.

c. The <*flexContainer*> for the Operation result. The attribute *notificationContentType* of the <*subscription*> needs to be set to "all-attributes".

5.3.2 Data flows for communicating with the IPE using DataPoints of a Service

The following figures show the data flows for communicating with the IPE using DataPoints of a Service.

-

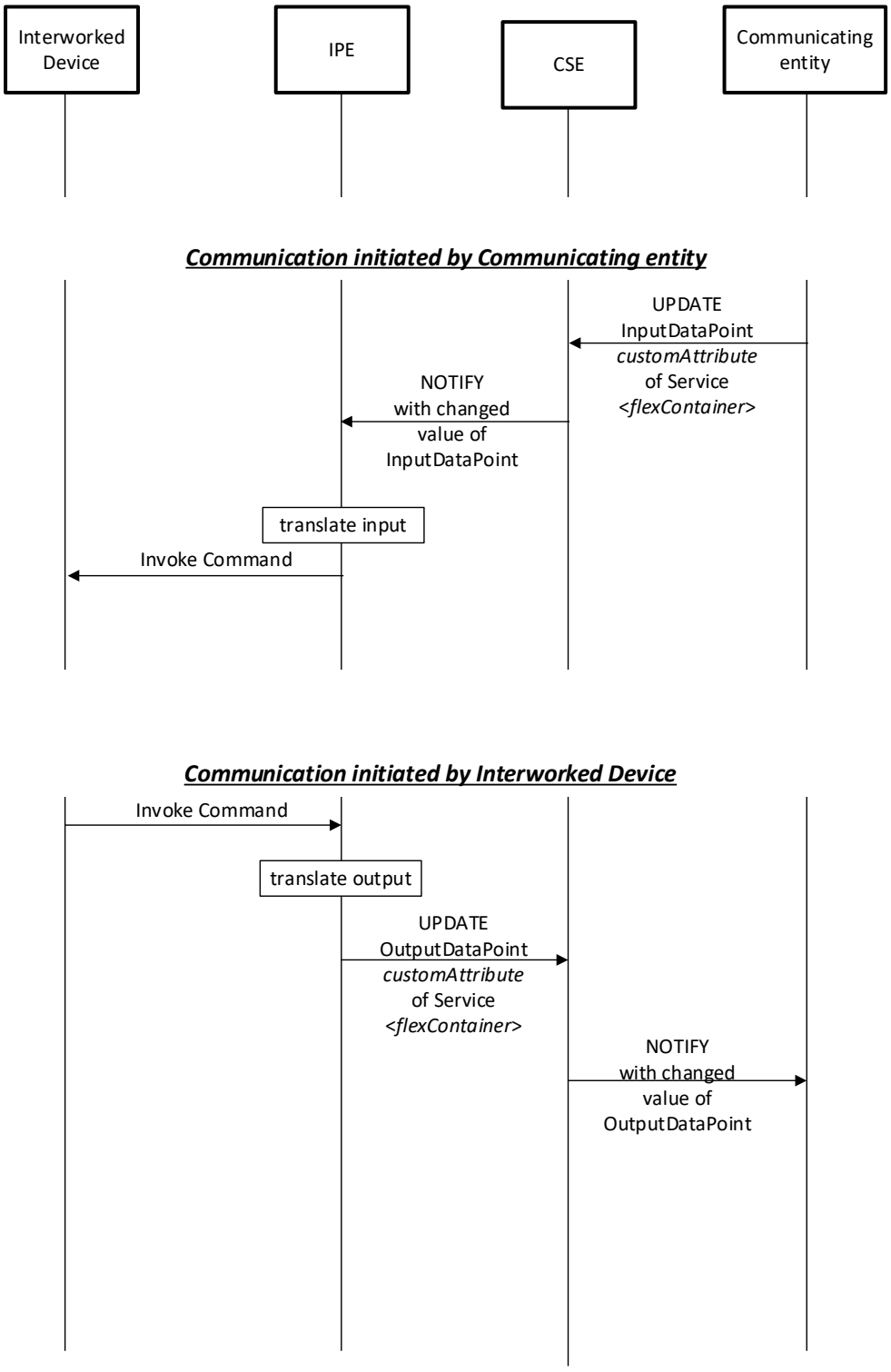

Figure 3: Data flow for an IPE involving dataPoints

When a communicating entity wants to invoke a command on the Interworked Device (e.g. an actuation command), using Datapoints of the related Service:

- The communicating entity UPDATEs the corresponding <*flexContainer*> that represents the service with the new value for *customAttribute* of the InputDatapoint.
- The CSE subsequently NOTIFYes the IPE about the changed value for *customAttribute* of the InputDatapoint.
- The IPE invokes the command at the Interworked Device that sends the data of the InputDatapoint to the Interworked Device.

-

When the Interworked Device wants to invoke a command (e.g. a reporting) on the IPE - or a subscribed communicating entity, using Datapoints of the related Service:

- The Interworked Device invokes the command at the IPE that sends the data of the OutputDatapoint to the IPE.
- The IPE UPDATEs the corresponding <*flexContainer*> that represents the Service with the new value for *customAttribute* of that OutputDatapoint.
- The CSE subsequently NOTIFYes subscribed communicating entities about the changed value for *customAttribute* of the OutputDatapoint.

5.3.3 Data flows for communicating with the IPE using Operations of a Service

The following figures show the data flows for communicating with the IPE using an Operation of a Service.

-

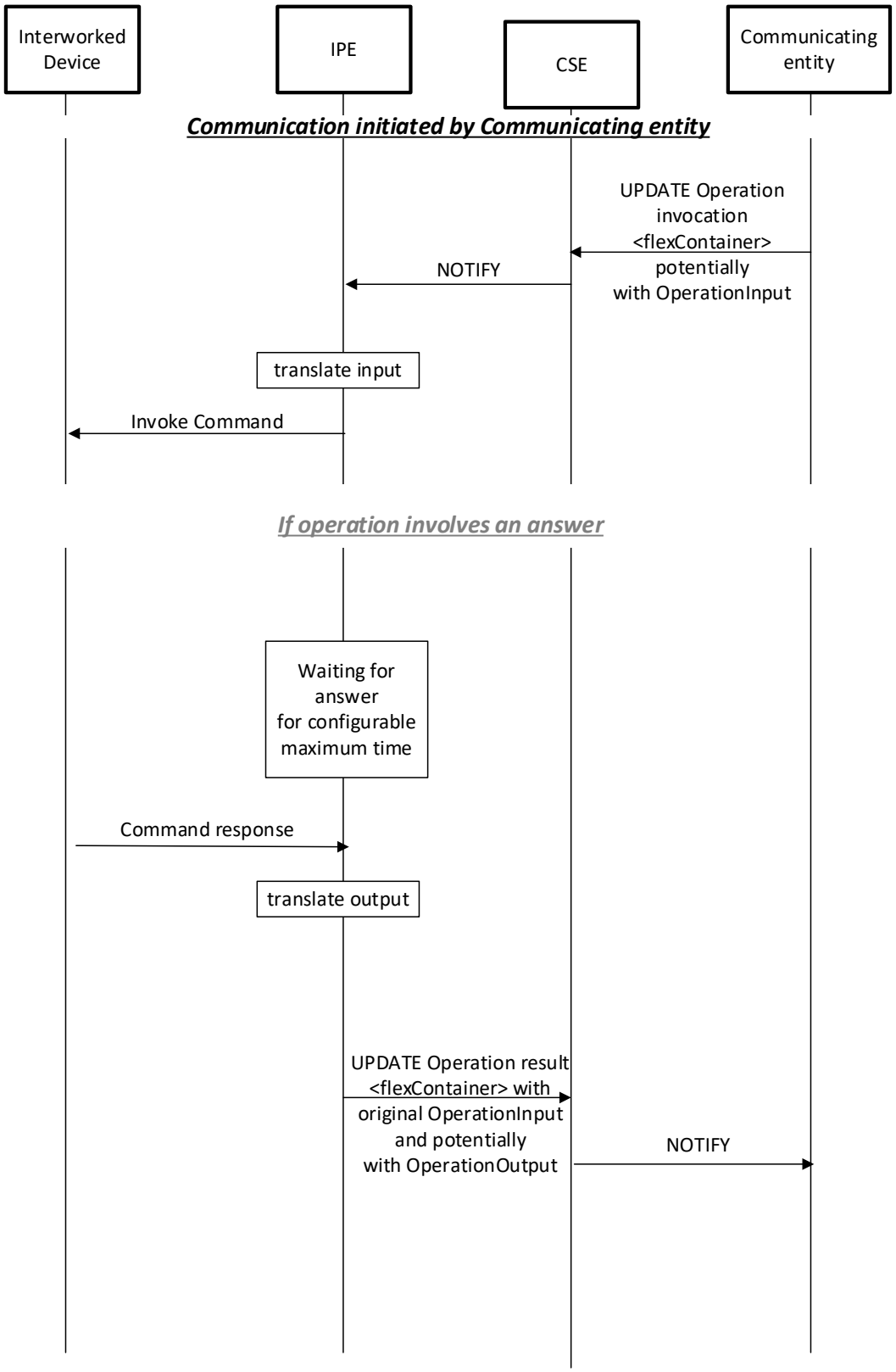

Figure 4: Data flow for a translating entity involving operations when initiated by a communicating entity

In contrast to DataPoints Operations allow grouping of input- and output parameters of a Command into a single transaction between the communicating entity and the target entity. It is permissible that no OperationInput and/or OperationOutput data exist for an Operation:

-

- When the communicating entity invokes an operation in the target entity it UPDATEs the <*flexContainer*> resource for the Operation invocation, potentially with OperationInput values for the related *customAttributes*. If the operation involves no OperationInput the UPDATE request contains no *customAttributes.*
- The CSE subsequently NOTIFYes the IPE about the updated <*flexContainer*> resource for the Operation invocation, potetially with OperationInput values for the related *customAttributes*.
- The IPE invokes the command at the Interworked Device.

If the Operation involves an (optional or mandatory) answer from the Interworked Device then:

- The Interworked Device returns the answer on the command, containing output data, to the IPE. The IPE needs to wait for that answer for a configurable maximum time that may depend on the technology of the interworked non-oneM2M solution.
- The IPE UPDATEs the corresponding <*flexContainer*> that represents the Operation result with OperationOutput values for the related *customAttributes.*
- The CSE subsequently NOTIFYes the communicating entity about the update of the <*flexContainer*> that represents the Operation result.

6 Functional specification of communication with the Ontology based Interworking IPE

6.1 oneM2M resources for IPE communication

6.1.1 General design principles

For Ontology based Interworking the oneM2M resource types <*AE*> and specializations of <*flexContainer*> are intended to hold data that can be used for data exchange with the IPE.

For Ontology based Interworking a convention is needed how the IPE uses these resources to communicate with other oneM2M entities. This is described in the subsequent clauses.

Resources for RESTful communication style vs. procedure call (RPC) style.

A Ontology based Interworking IPE needs to be able to communicate with systems that implement some form of RESTful communication style as well as other systems that communicate in a procedure call (RPC) style.

For RESTful systems the use of Input- or OutputDataPoints may be more appropriate.

On the other hand procedure calls that involve stateful transactions between the IPE and the Interworked Device can be better modelled using Operations (and their OperationInputs/-Outputs).

6.1.2 Resource structure for modelling devices, sub-devices, services and operations

Figure 5 and Figure 6 provide an overview of parent-child resource relationships that are used for communication with Interworked Devices in the context of Ontology based Interworking.

It involves the oneM2M resource types:

- <*AE*> - for the Interworking Proxy Entity (IPE).
- A <*container*> child resource of the <*AE*> of the IPE - for holding XSD files (for the specializations of <*flexContainers*> used in the interworking.

-

Interworked Devices are represented either by:

- <*flexContainer*> - child- resource of <*AE*> of IPE; or
- <*AE*> - child resources of the hosting CSE of the IPE.

NOTE: An IPE may, instead of creating <*flexContainer*>resources for Interworked Devices, also choose to represent them as <*AE*>s. This option should be chosen if the interworked devices need to be identifiable for the purpose of service subscription, charging, differentiation during access control enforcement, authentication, App-ID registry, etc.

- <*flexContainer*> - for a sub-device
  (child-resource of the <*flexContainer*> or <*AE*> of its parent device).
- <*node*> for the node of a Device or sub-device.
- <*flexContainer*> - for a Service of a Device or sub-device.
  (child-resource of the <*flexContainer*> or <*AE*> of the device that offers the Service).
  - <*flexContainer*>s for network services (e.g. broadcast services, registration services …) that are offered by the IPE are child-resources of the IPE's <*AE*>.
- <*flexContainer*> - for an Operation (invocation) of a Service.
  (child-resource of the <*flexContainer*> of the Service).
- <*flexContainer*> - for an Operation (result) of a Service.
  (child-resource of the <*flexContainer*> of the Operation invocation of the Service).

-

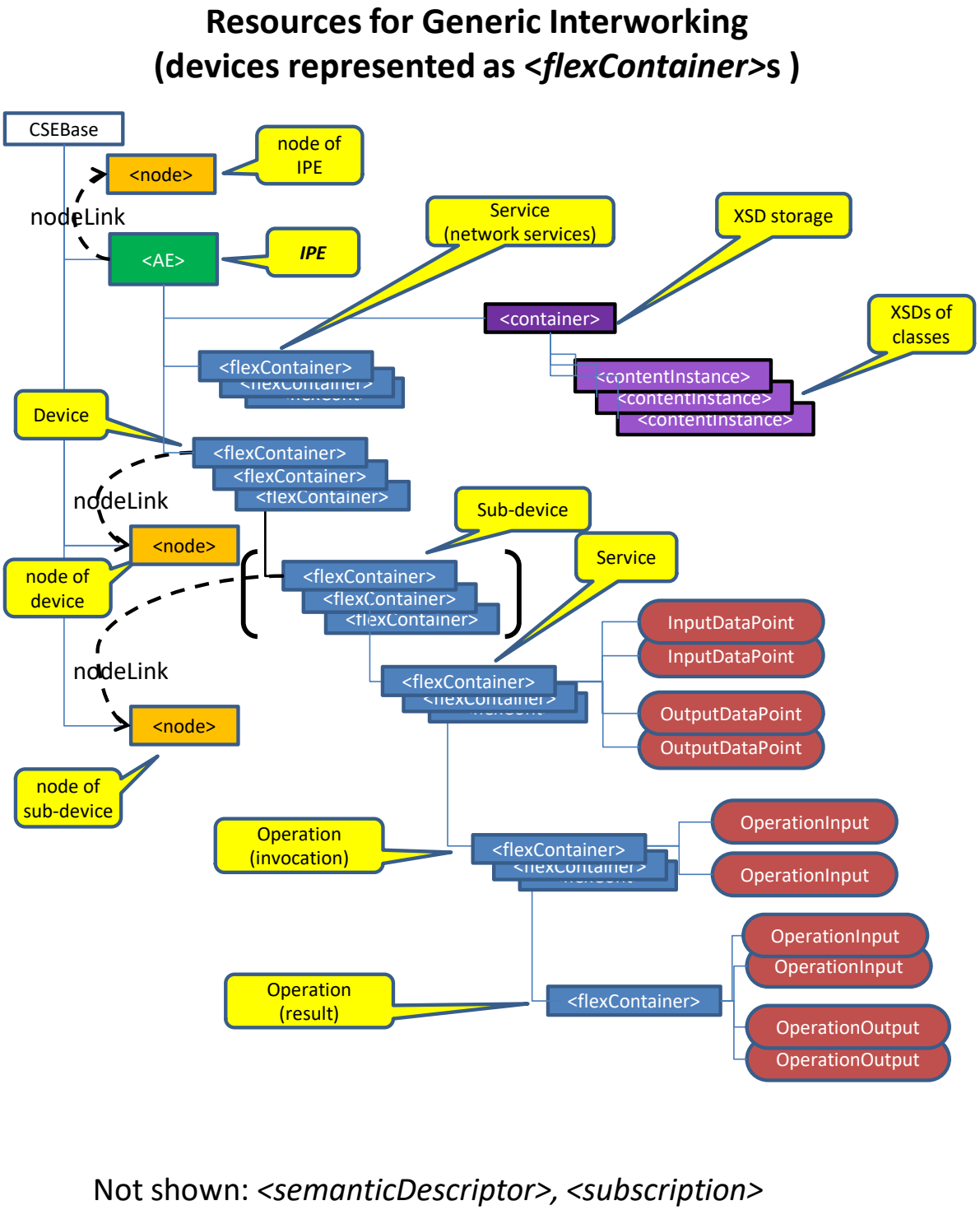

Figure 5: Resources used in the context of Ontology based Interworking when Interworked Devices are represented as <*flexContainer*>s

Parent-child relationships when Interworked Devices are represented as <*flexContainer*>s:

- A <*container*> resource, for holding XSD files, is created as child-resource of the <*AE*> resource of the IPE.
- <*flexContainer*> resources, for representing (network) Services are created as child-resources of the <*AE*> of the IPE.

-

- <*flexContainer*> resources for representing Interworked Devices are created by the Ontology based Interworking IPE as child-resources of the IPE's <*AE*>.
- <*flexContainer*> resources, for representing sub-devices are created as child-resources of the <*flexContainer*> resource that represents the device to which the sub-devices belong.
- <*flexContainer*> resources, for representing Services are created as child-resources of the <*flexContainer*> resource that represents the device or sub-device to which the Service belongs. Input- or OutputDataPoints are modelled as *customAttributes.*
- <*flexContainer*> resources, for representing Operations invocations are created as child-resources of the <*flexContainer*> resource that represents the Service to which the Operation belongs.
  Only OperationInputs are modelled as *customAttributes.*
- <*flexContainer*> resources, for representing Operations results are created as child-resources of the <*flexContainer*> resource that represents the Operations invocation to which the Operation result belongs.
  Both, OperationInputs and -Outputs are modelled as *customAttributes.*
- All of the above can contain a <*semanticDescriptor*> as child resource and may contain <*subscription*> resources.

-

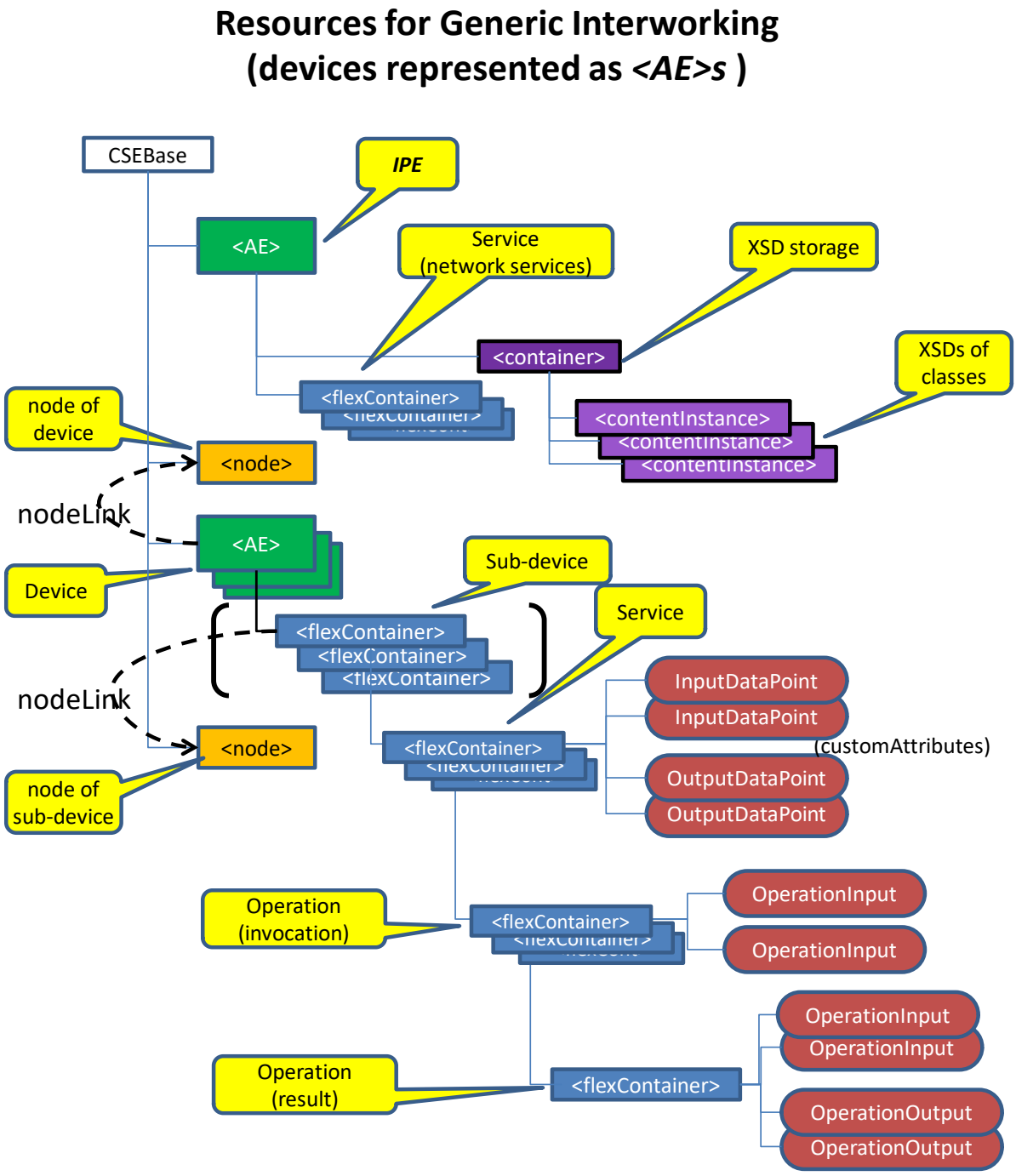

Figure 6: Resources used in the context of Ontology based Interworking when Interworked Devices are represented as <*AE*>s

-

Parent-child relationships when Interworked Devices are represented as <*AE*>s:

- A <*container*> resource, for holding XSD files, is created as child-resource of the <*AE*> resource of the IPE.
- <*flexContainer*> resources, for representing (network) Services are created as child-resources of the <*AE*> of the IPE.
- An <*AE*> resource, representing each Interworked Device is created by the IPE as child-resources of the IPE's hosting CSE.
- <*flexContainer*> resources, for representing sub-devices are created as child-resources of the <*AE*> resource that represents the device to which the sub-devices belong.
- <*flexContainer*> resources, for representing Services are created as child-resources of the <*AE*> that represents the device (or <*flexContainer*> that represents the sub-device) to which the Service belongs. Input- or OutputDataPoints are modelled as *customAttributes.*
- <*flexContainer*> resources, for representing Operation invocations are created as child-resources of the <*flexContainer*> resource that represents the Service to which the Operation belongs. Only OperationInputs are modelled as *customAttributes.*
- <*flexContainer*> resources, for representing Operation results are created as child-resources of the <*flexContainer*> resource that represents the Operations invocation to which the Operation result belongs. Both, OperationInputs and -Outputs are modelled as *customAttributes.*
- All of the above can contain a <*semanticDescriptor*> as child resource and may contain <*subscription*> resources.

Link relationships:

- The <*AE*>s and <*flexContainer*> resources, representing Interworked Devices / fs may have *nodeLink* attributes to their respective <*node*> resources.

NOTE: If Interworked Devices / sub-devices are resident on the same node their *nodeLink* attributes reference the same <*node*> resource.

6.2 Specification of the IPE for Ontology based Interworking

6.2.1 Initialization of the Ontology based Interworking IPE

6.2.1.1 General functionality of a Ontology based Interworking IPE

Ontology based Interworking Interworking supports the interworking variant with full mapping of the semantic of the non-oneM2M data model to Mca as indicated in clause F.2 of oneM2M TS-0001 [2].

The non-oneM2M data model is described in the form of a oneM2M compliant ontology which is derived (as sub-classes and sub-properties) from the oneM2M Base Ontology and may be available in a formal description language (e.g. OWL).

A oneM2M compliant ontology can describe an external technology (e.g. ZigBee) for which a standardized interworking with oneM2M is required or it could describe a model of consensus that is shared by large industry sector (like SAREF, referenced in [i.2]) that facilitates the matching of existing assets (standards/protocols/datamodels, etc.). An IPE that provides Ontology based Interworking with a non-oneM2M system (e.g. an M2M Area Network) shall instantiate the classes, object- and data properties of the ontology describing the non-oneM2M data model of the M2MArea Network as oneM2M resources, according to the instantiation described below.

In the following clauses it is assumed that the oneM2M compliant ontology describing the non-oneM2M data model is available as a formal description (e.g. in OWL format). The location (URI) where the formal description of the ontology can be retrieved by the IPE needs to be configured in the IPE - either preconfigured or through administrative means.

-

6.2.1.2 Initialization sequence of a Ontology based Interworking IPE

After registration of the IPE's <*AE*> resource the IPE shall do the following:

1) The IPE shall retrieve the formal description of the ontology and parse it.

2) The IPE shall create a child-resource of its <*AE*> resource of type <*container*>, subsequently called 'XSD-storage'
This <*container*>is will hold the XSDs that are needed to specialize the <*flexContainer*>s that are used in the oneM2M resource representation of entities of the non-oneM2M interworked solution:

   - The resource name of the <*container*> resource shall be identical to the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).

   - The attribute *ontologyRef* of that <*container*>shall contain the reference (URI) to the used ontology.

3) For each class of the ontology the IPE shall create a resource of type <*contentInstance*> as child resource of this <*container*>. That <*contentInstance*> shall hold the XSD for instantiating the class as <*flexContainer*> specialization in oneM2M.
The specification how the IPE shall create XSDs from an ontology is described in clause 7.

   - The resource name of each <*contentInstance*> resource shall be identical to the class name of the ontology, followed by ".xsd".

   - The attribute *ontologyRef* of that < *contentInstance* >shall contain the reference (URI) to the class of the ontology.

EXAMPLE:

For ontology 'http://www.someOrganization.org/someOntology' the IPE would create a <container> resource with resource name "www.someOrganization.org_someOntology".
Its *ontologyRef* would contain the URI: 'http://www.someOrganization.org/someOntology'.

If the ontology defines a class 'http://www.someOrganization.org/someOntology#someService' then the IPE needs to create a <*contentInstance*> as child rsource of that that "www.someOrganization.org_someOntology" <*container*>. The resource name of that <*contentInstance*> would be "someService.xsd".
Its *ontologyRef* would contain the URI:
'http://www.someOrganization.org/someOntology#someService'.

4) If supported by the technology of the non-oneM2M solution the IPE shall create <*flexContainer*>s according to clause 6.2.2.4 for its own (network) services (e.g. broadcast services…) that are provided by the IPE.

5) If supported by the technology of the non-oneM2M solution the IPE shall discover the devices in the non-oneM2M solution, including their supported services. Alternatively, information about the devices in the the non-oneM2M solution may be manually configured in the IPE or obtained by other means.

6.2.2 Interworked Device and Service discovery

6.2.2.1 General handling of Interworked Device discovery

If supported by the technology of the non-oneM2M solution the IPE shall continue to discover the devices in the non-oneM2M solution.

If supported by the technology of the non-oneM2M solution the IPE shall continue to discover services that are provided by the devices in the non-oneM2M solution.

-

6.2.2.2 Creation of resources for the Proxied Device

6.2.2.2.1 General on the creation of resources for the Proxied Device by the IPE

For each discovered Interworked Device in the non-oneM2M solution the IPE shall:

either:

- create a <*flexContainer*>child-resource of the IPE's own <*AE*> to represent the non-oneM2M Interworked Device in the oneM2M System.
  - This option should be chosen in most cases, in particular when the individual representations for Interworked Devices - the Proxied Devices - do not need to be distinguished with regard to service subscription, charging, etc. as these are derived from the IPE'S <*AE*>.

or

- register an <*AE*> resource with the IPE's hosting CSE. That <*AE*> resource represents the non-oneM2M Interworked Device in the oneM2M System.
  - This option should be chosen if the interworked devices need to be identifiable for the purpose of service subscription, charging, differentiation during access control enforcement, authentication, App-ID registry, etc.

NOTE: The resource (<*flexContainer*> or <*AE*>) that represents the non-oneM2M Interworked Device in the oneM2M System, together with its child resources, and together with the IPE functionality to execute CRUDN operations on these resources is called 'Proxied Device' (see definitions clause).

6.2.2.2.2 Creation of resources for the Proxied Device when Interworked Devices are represented as <*flexContainer*>s

If the IPE creates <*flexContainer*>s for Proxied Devices the following rules apply.

For each discovered device in the non-oneM2M solution the IPE shall create a specialized <*flexContainer*> resource as child-resource of the <*AE*> resource of the IPE.

- The specialization type of the <*flexContainer*> is determined by the XSD file - a <*contentInstance*>in the XSD-storage - that correlates to the class of the sub-device in the ontology.
- It is recommended that the resourceName of an <*flexContainer*> resource that represents an Interworked Device should be derived and resemble the address (e.g. MAC address) of the Interworked Device in the non-oneM2M solution.
  As the formats of such addresses are very diverse no general rule for that derivation can be given.
- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the Interworked Device.
- The labels attribute of the <*flexContainer*> may contain the following key- vale pairs:
  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).
  - Key: "Iwked-Entity-Type" Value: Class name of the class in the ontology (i.e. a sub-class of class InterworkedDevice of the oneM2M Base OntologyBase Ontology) that specifies the type of the Interworked Device
- The *resourceID* of a <*node*> resource that stores the node specific information where this Interworked Device resides may be contained in the *nodeLink* attribute of the <*flexContainer*> of the Device.

NOTE: Such a <*node*> resource is required if the device management tasks can be performed via the oneM2M solution.

- The IPE may create a <*semanticDescriptor*> child-resource for the <*flexContainer*> resource that represents an Interworked Device (see clause 7 in [3]).

-

6.2.2.2.3 Creation of resources for the Proxied Device when Interworked Devices are represented as <*AE*>s

If the IPE registers individual <*AE*>s for Proxied Devices the following rules apply:

- As the IPE registers the <*AE*> as a proxy for the Interworked Device it should either:
  - not create a Security Association Establishment procedure, or
  - create a Security Association Establishment procedure between the Node on which the IPE <AE> is hosted and the Registrar CSE. In this case only the Node from which the registration request is received at the Registrar CSE is authenticated. Thus the IPE, which handles communication for all the Proxied Devices, can communicate over either a single Security Association or over individual Security Associations for each Proxied Device.

NOTE 1: The Node authentication as described above (see also oneM2M TS-0001 [1] clause 10.2.2.2) is applicable as the IPE performs AE functionality for all its Proxied Devices, as if their (virtual) individual AEs were resident on the same node as the IPE AE.

- The APP-ID of the <*AE*> shall be the APP-ID of the IPE.
- It is recommended that the AE-ID of an <*AE*> resource that represents an Interworked Device should be derived and resemble an identifier of the Interworked Device in the non-oneM2M solution.
  As the formats of such identifiers are very diverse no general rule for that derivation can be given.
- It is recommended that the resourceName of an <*AE*> resource that represents an Interworked Device should be derived and resemble the address of the Interworked Device in the non-oneM2M solution.
  As the formats of such addresses are very diverse no general rule for that derivation can be given.
- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the Interworked Device.
- The labels attribute of may contain the following key- vale pairs:
  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).
  - Key: "Iwked-Entity-Type" Value: Class name of the class in the ontology (i.e. a sub-class of class InterworkedDevice of the oneM2M Base OntologyBase Ontology) that specifies the type of the Interworked Device.
- The *resourceID* of a <*node*> resource that stores the node specific information where this Device resides may be contained in the *nodeLink* attribute of the <AE> of the Device.

NOTE 2: Such a <node> resource is required if the device management tasks can be performed via the oneM2M solution.

- The IPE may create a <*semanticDescriptor*>child-resource for the <*AE*> (see clause 7 in [4]).

6.2.2.3 Creation of resources for sub-devices

A Device can consist of (i.e. be composed) of several (sub-) Devices. In the oneM2M Base OntologyBase Ontology the Object Property: consistsOf links a device class to a class of its sub-devices.

For each sub-device of a discovered device in the non-oneM2M solution the IPE shall create a specialized <*flexContainer*> resource as child-resource of the <*AE*> or <*flexContainer*> resource of the Proxied Device/:

- The specialization type of the <*flexContainer*> is determined by the XSD file - a <*contentInstance*>in the XSD-storage - that correlates to the class of the sub-device in the ontology.
- The resourceName of the <*flexContainer*> resource of the sub-device shall be identical to the class name of the (ontology specific sub-class of) class:Device of the sub-device in the ontology.
  Example: "switchingSubDevice".

-

- If multiple sub-device instances of the same class exist then the resourceName shall be appended by '_' (underline) and followed by a number to distinguish individual <*flexContainer*> resources for multiple (sub-)devices of the same type.
  Example: "subDeviceCuff _02" when the device is a blood pressure monitor containing 4 individual cuffs for measuring blood pressure as sub-devices.

NOTE: The creation of <*flexContainer*> resources for sub-devices of a device is basically analogous to the creation of <*flexContainer*> resources for Services of a device (see next clause).

- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the sub-device.

- The labels attribute of may contain the following key- vale pairs:

  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).

  - Key: "Iwked-Entity-Type" Value: Class name of the class of the sub-device in the ontology (i.e. a sub-class of class:InterworkedDevice of the oneM2M Base OntologyBase Ontology).

- The IPE may create a <*semanticDescriptor*>child-resource.

6.2.2.4 Creation of resources for Services

For each service, that is supported by a device or sub-device the IPE shall create a specialized <*flexContainer*> resource, representing the service. The <*flexContainer*> shall be a child-resource of the Proxied Device representation (<*AE*> or <*flexContainer*> resource in case of an interworked Device, specialized <*flexContainer*> in case of a sub-device):

- The services that can be supported by a device or sub-device are specified in the ontology by Object Property: hasService:

  - If a service is mandatory for a specific device then Object Property:hasService (or a sub-property) of the ontology has a Restriction on cardinality: "exactly 1".
    In this case the IPE shall create the specialized <*flexContainer*> for the Service.

  - If a service is optional for a specific device and can have only one instance on the device then Object Property:hasService (or a sub-property) has a restriction on cardinality: "max 1".
    In this case the IPE shall create the specialized <*flexContainer*> for the Service only if the the IPE can determine (using methods of the interworked solution) that the (sub)device actually supports the service.

  - A service can have multiple instances on a device. In this case Object Property:hasService (or a sub-property) has no restriction on cardinality or a restriction: e.g. "min *[x]*"with x>1:

    - The ontology may specify that the same service of a device exposes multiple, different Functions (Object Property:exposesFunction). In this case multiple instances of the same service might be distinguishable by the Function they expose.

    - If the ontology does not specify the Function that is exposed by the Service or if multiple Services with the same Function exist in the device then these are semantically indistinguishable
      Example: a connector strip containing 5 individual switching services.

    In this case the IPE may create multiple specialized <*flexContainer*> instances for the Service.

- The specialization type of the <*flexContainer*> is determined by the XSD file - a <*contentInstance*> in the XSD-storage - that correlates with the class of the service in the ontology.

NOTE: The XSD file for the service also includes XSD descriptions for the service's DataPoints - which are represented as *customAttribute*s of the<*flexContainer*>for the Service.

- The resourceName of the <*flexContainer*> resource of the service shall be identical to the class name of the (ontology specific sub-class of) class:Service in the ontology.
  Example: "liquidRemaining".

-

  - If multiple service instances of the same class:Service exist but expose different Functions (via Object Property:exposesFunction) then the resourceName shall be appended by '_' (underline), followed by the class name of the function in the ontology.
    Example: "liquidRemaining_ waterStatus", "liquidRemaining_ milkStatus".

  - If multiple service instances of the same class:Service exist that expose the same function then the resourceName shall be appended by '_' (underline), followed by a number to distinguish individual <*flexContainer*> resources for multiple services of the same type.
    Example: "liquidRemaining_ waterStatus_01", "liquidRemaining_ waterStatus_02".

- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the service.

- The labels attribute of may contain the following key- vale pairs:

  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).

  - Key: "Iwked-Entity-Type" Value: Class name of the class of the service in the ontology (i.e. a sub-class of class:Service of the oneM2M Base Ontology).

- The IPE may set access rights for the <*flexContainer*> resource of the service to restrict/allow AEs to use (UTDATE and / or RETRIEVE and SUBSCRIBE to) the the <*flexContainer*> that represents the Service.

- The IPE may create a <*semanticDescriptor*>child-resource of the <*flexContainer*> resource of the service.

6.2.2.5 Creation of resources for operations of a service of a device

6.2.2.5.1 Introduction

An Operation is the means of a Service to communicate in a procedure-type manner.

In general an operation is invoked by the communicating entity and can - but need not - have OperationInput parameters. As a result of the Operation result parameters can - but need not - be returned.

An operation can also be invoked by the Interworked Device and can - but need not - have OperationOutput parameters. In this case no operation results are envisaged.

NOTE: Analogous to operation invocation by the communicating entity at the Interworked Device also operation invocation by the Interworked Device at another device (oneM2M native or interworked) can happen. This case is not supported in the current release.

Operations are represented in the oneM2M systems with one or two types of resources, both of them being specialized <*flexContainer*> resources.

- The first type exists for every operation and is used to invoke the operation:

  - In the case the operation is invoked by the communicating entity it only contains OperationInput parameters, if they exist for this type of operation.

  - In the case the operation is invoked by the Interworked Device it only contains OperationOutput parameters, if they exist for this type of operation.

- The second type is a child-resource of the first type and only exists if the operation had been invoked by the communicating entity and only for operations that can produce OperationOutput parameters. This type contains both, the OperationInput parameters, if they exist, and the OperationOutput parameters, if they exist.

6.2.2.5.2 Creation of resources for operation invocation

For each operation, that is supported by a service of a device the IPE shall create a specialized <*flexContainer*> resource, representing the operation. The <*flexContainer*> shall be a child-resource of the specialized <*flexContainer*> of the service.

-

The IPE shall subscribe to all specialized <*flexContainer*> resources representing operations, that it creates.

- The operations that can be supported by a service are specified in the ontology by Object Property:hasOperation.

  - An Operation for a specific Service can only have at most a single instance at a time. Unless the operation is optional, i.e. Object Property:hasOperation (or a sub-property) has a restriction on cardinality: "max 1" the IPE shall create the specialized <*flexContainer*> that represents the Operation.

NOTE 1: It is quite well possible, that the Interworked Device supports execution of multiple operation instances of the same operation type at a time. However, mutiple invocations of the same operation type is achieved in oneM2M by consecutively UPDATEing a single specialized <*flexContainer*> resource that represents the Operation input. Thus a single <*flexContainer*> resource is sufficient to invoke multiple, concurrent operation instances of that operation in the Interworked Device.
Similarly, also returning OperationOutput parameters is done through a single UPDATE operation on the specialized <*flexContainer*> resource that represents the Operation output.

  - If an operation is optional for a specific service then Object Property:hasOperation (or a sub-property) has a restriction on cardinality: "max 1".
    In this case the IPE shall create the specialized <*flexContainer*> for the Operation only if the the IPE can determine (using methods of the interworked solution) that the Service actually supports the operation.

- The specialization type of the <*flexContainer*> is determined by the XSD file - a <*contentInstance*> in the XSD-storage - that correlates with the class of the operation in the ontology.

NOTE 2: The XSD file for the operation also includes XSD descriptions for the operation's OperationInput parameters which are represented as *customAttribute*s of the<*flexContainer*>for the operation.
When the IPE creates the <*flexContainer*> resource it may ignore *customAttribute*s that represent the operation's OperationOutput parameters.

- The resourceName of the <*flexContainer*> resource of the operation shall be identical to the class name of the (ontology specific sub-class of) class:Operation in the ontology.
  Example: "toggle" as an operation of a service of a light switch, "upVolume" as an operation of a service of a audio device.

- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the operation.

- The labels attribute of may contain the following key- vale pairs:

  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).

  - Key: "Iwked-Entity-Type" Value: Class name of the class of the operation in the ontology (i.e. a sub-class of class:Operation of the oneM2M Base Ontology).

- The IPE may set access rights for the <*flexContainer*> resource of the service to restrict/allow AEs to use (UTDATE and / or RETRIEVE and SUBSCRIBE to) the the <*flexContainer*> that represents the Operation.

- The IPE may create a <*semanticDescriptor*>child-resource of the <*flexContainer*> resource of the Operation.

6.2.2.5.3 Creation of resources for returning operation results

For each operation, that can produce operation result parameters (i.e. its class:Operation is related via Object Property: hasOutput to some class:OperationOutput) the IPE shall create a specialized <*flexContainer*> resource, representing the result of the operation for a set of invocation parametes. The <*flexContainer*> shall be a child-resource of the specialized <*flexContainer*> resource for operation invocation (described in clause 6.2.2.3.2).

The IPE shall subscribe to this specialized <*flexContainer*> resource:

- The specialization type of the <*flexContainer*> is determined by the XSD file - a <*contentInstance*> in the XSD-storage - that correlates with the class of the operation in the ontology.

-

NOTE: The XSD file for the operation also includes XSD descriptions for the operation's OperationInput and OperationOutput parameters which are represented as *customAttribute*s of the<*flexContainer*>.

- The resourceName of the <*flexContainer*> resource of the operation shall be identical to the class name of the (ontology specific sub-class of) class:Operation in the ontology, appended by '_result'.
  Example: "upVolume_result" as the result of an operation that e.g. returns the current volume level as OperationOutput parameter.

- The attribute *ontologyRef* shall contain the reference (URI) to the class of the ontology that specifies the type of the operation.

- The labels attribute of may contain the following key- vale pairs:

  - Key: "Iwked-Technology" Value: the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore).

  - Key: "Iwked-Entity-Type" Value: Class name of the class of the operation in the ontology (i.e. a sub-class of class:Operation of the oneM2M Base Ontology).

- The IPE may set access rights for RETRIEVEing the <*flexContainer*> resource of the operation.

- The IPE may create a <*semanticDescriptor*>child-resource of the <*flexContainer*> resource of the operation.

6.2.2.6 Subscription to the created resources

The IPE shall subscribe to all specialized <*flexContainer*> resources representing services, that it creates.
In particular it shall subscribe to the <*flexContainer*> resources representing Services and <*flexContainer*> resources for the Operation invocation. The attribute *notificationContentType* of the <*subscription*> needs to be set to "modified-attributes".

6.2.3 Handling of DataPoints by the IPE

- When the IPE receives a request by the interworked non-oneM2M device via the non-oneM2M reference point to write an OutputDataPoint belonging to a Service of the device the IPE shall:

  - de-serialize the received data; and

  - UPDATE the OutputDataPoint *customAttribute* of the <*flexContainer*> resource of the Service with the output data.

- When the IPE receives a request by the interworked non-oneM2M device via the non-oneM2M reference point to read an InputDataPoint belonging to a Service of the device the IPE shall:

  - RETRIEVE data from the InputDataPoint *customAttribute* of the <*flexContainer*> resource of the Service;

  - serialize the data; and

  - return them to the non-oneM2M device.

- When the IPE is notified by the CSE that a *customAttribute* of the <*flexContainer*> resource of the Service of the Proxied Device has been changed the IPE shall:

  - read the data of the changed *customAttribute;* and

  - invoke the Service, parameterized with data of the InputDataPoint, via the non-oneM2M reference point in the interworked non-oneM2M device.

-

6.2.4 Handling of Operations by the IPE

When the IPE receives notification from the CSE about an UPDATE of the <*flexContainer*> resource of the operation the IPE shall perform the following actions:

1. If the operation has OperationInput data the IPE shall RETRIEVE the OperationInput data of the operation (contained in the *customAttribute*s of the <*flexContainer*> resource of the operation).

2. The IPE shall serialize OperationInput data and invoke the related operation in the non-oneM2M device via the non-oneM2M reference point.

3. If the operation allows OperationOutput data (i.e. a <*flexContainer*> resource for returning operation results exists as a child-resource of the operation <*flexContainer*>) the IPE shall handle the result of the operation, when received from the Interworked Device via the non-oneM2M reference point:

    - The IPE shall de-serialize OperationOutput data.

    - The IPE shall UPDATE the <*flexContainer*> resource for returning operation results. The UPDATE primitive shall contain:

        - the *customAttribute*s with the values for the OperationInput data with which the operation had been invoked, and

        - the *customAttribute*s with the values for the OperationOutput data that had been received as result of the operation.

NOTE: The IPE needs to await the operation results by the Interworked Device for an operation transaction only for a configurable timespan that may depend on the technology of the non-oneM2M solution. The duration of that timespan is out of scope of the current specification.

When the the non-oneM2M device invokes an operation in the IPE - which may or may not contain OperationInput data - via the non-oneM2M reference point (e.g. when the device reacts on some external event and publishes related output data):

1. The IPE shall de-serialize these data and perform the following actions:

    - If the operation invocation contained no OperationInput data the IPE shall UPDATE the <*flexContainer*> resource for the operation with NULL values for all *customAttribute*s.

    - If the operation invocation contained OperationOutput data then:

        - The IPE shall de-serialize OperationOutput data.

        - The IPE shall UPDATE the OperationOutput *customAttribute*s of the <*flexContainer*> resource for the operation with the values received.

6.2.5 Removing of resources for Proxied Devices

- When a Interworked Device in the non-oneM2M solution becomes unavailable the IPE shall delete the resource for its Proxied Device and all its child-resources.

- When the IPE detects that an Interworked Device stopped to support a Service (in case such detection is supported by the technology of the non-oneM2M solution) then the IPE shall delete the <*flexContainer*> resource for the service and all its child-resources.

-

7 Rules for creation of XSDs from ontologies

7.1 General information

When the non-oneM2M data model is described in the form of a oneM2M compliant ontology which is derived (as sub-classes and sub-properties) from the oneM2M Base Ontology and is available in a formal description language (e.g. OWL) then the IPE can create XSDs for the resources needed by communicating entities to communicate with the IPE.

These resources are specializations of <*flexContainer*>*s* for Devices, Services and Operations. For these specializations of <*flexContainer*>*s* the IPE also needs to create XSD definitionss for *customAttributes* of the <*flexContainer*>.

NOTE: The alternative of using <*AE*> resources for Devices is not used for Ontology based Interworking as specified in the current document. <*AE*> resource types are described in oneM2M TS-0001 [2] and oneM2M TS-0004 [8].

7.2 XSD creation rules

7.2.1 General rules

7.2.1.1 General principle for creating XSDs

During initialization (clause 6.2.1) a Ontology based Interworking IPE creates XSD files for device types, service types and operation types, according to the class definitions (sub-classes of class:Device, class:Service, class:Operation) in the ontology. Each XSD file contains the definition of a single specialization of a <*flexContainer*> resource type together with the type definitions that are used for *customattribute*s of that specialization. All of these XSD files are stored as <*contentInstance*> child- resources of a <*container*> resource that acts as a XSD-storage and that is a child resource of the IPE's <*AE*>.

The content of these XSD files is entirely based on the information contained in the oneM2M compliant ontology that describes the data model of the interworked technology and which is derived from the oneM2M Base Ontology.
A Ontology based Interworking IPE can create these XSD files automatically.

In clauses 7.2.2, 7.2.3, 7.2.4 templates for specialization of <*flexContainer*> resources for Device, Service and Operation are given. These templates need to be filled in with parameters. The parameters and how they relate to classes, object properties and data properties of the ontology are given in the next clause 7.2.1.2.

The attribute: *containerDefinition* of these XSDs shall have a value identical to the absolute, hierarchical address of the <*contentinstance*> containing the <*flexContainer*> specialization (see table 1 below).
Example: //m2m.service.com/IN-CSE-0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines /testService.xsd

NOTE: The present specification does not specify short names for primitive parameters, resource attributes, resource types and complex data types members. Such short names can be created on a proprietary basis for individual implementations but for interoperability reasons they are not recommended.

7.2.1.2 Parameters for XSD templates

In the subsequent clauses the following conventions for parameters to fill in XSD templates is used. The column "replacement rules" explains how the parameter is derived from the ontology.

NOTE: This convention follows roughly conventions of ENTITY declarations for DTD. Each ENTITY that is to be replaced in the XSD starts with "&" and ends with";".

-

Table 1: Conventions for creation of XSDs from ontologies

-

convention	Is short for	replacement rule
&IPE;	resource ID of the <*AE*> resource of the IPE	Out of scope of the present document.
&XSDSTORAGE;	resource name of the <*container*> resource of the 'XSD-storage'. also identifies the XML targetnamespace: targetNamespace="&XSDSTORAGE;"	Shall be identical to the name of the ontology, omitting "http://" and changing each "/" (slash) into "_" (underscore). Base OntologyThe 'XSD-storage' <*container*> is a child-resource of the <*AE*> resource of the IPE and has <*contentinstance*> child resources that contain XSDs for the individual <*flexContainer*> specializations. Example for &XSDSTORAGE;: ww.XYZ.com_WashingMachines
&XSDFILE;	resource name of a <*contentInstance*> child-resource of the 'XSD-storage'that contains a XSD for a <*flexContainer*> specialization.	Shall be the concatenation of • &DEVICE; or • &SERVICE; or • &OPERATION; and ".xsd" Example: "testService.xsd"
&CONTAINERDEFINITIONVALUE;	The value of attribute *containerDefinition* of the specialization of the <*flexContainer*>	Shall be identical to the absolute, hierarchical address of the <*contentinstance*> containing the <*flexContainer*> specialization. It is the concatenation of: [address of the Gen-IWK IPE], "/", &XSDSTORAGE;,"/", &XSDFILE; Example: //m2m.service.com/IN-CSE-0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines/testService.xsd
&DEVICE; &DEVICE_1; … &DEVICE_n;	Name of <*flexContainer*> specialization type for a device or a sub-device.	Shall be identical to a class name of a class:Device that is in the range of object property:consistsOf. It identifies a device type. Base Ontology
&SERVICE_1; … &SERVICE_n;	Name of <*flexContainer*> specialization type for a Service	Shall be identical to the name of a Service class in the ontology. The class is in the range of object property:hasService of a Device. (See Note 3)
&OPERATION_1; … &OPERATION_n;	Name of <*flexContainer*> specialization type for an Operation ***See Note 1***	Shall be identical to the name of an Operation class in the ontology The class is in the range of object property:hasOperation of a Service.
&THINGPROPERTY_1; … &THINGPROPERTY_n;	Name of a *customAttribute* of the Device's <*flexContainer*> describing a ThingpProperty of the device.	Shall be identical to the name of a ThingProperty class in the ontology The class is in the range of object property: hasThingProperty of a Device
&INPUTDATAPOINT_1; … &INPUTDATAPOINT_n;	Name of a *customAttribute* of the Service's <*flexContainer*> describing an InputDataPoint	Shall be identical to the name of a InputDataPoint class of the service The class is in the range of object property:hasInputDataPoint of a Service.
&OUTPUTDATAPOINT_1; … &OUTPUTDATAPOINT_n;	Name of a *customAttribute* of the Service's <*flexContainer*> describing an OutputDataPoint	Shall be identical to the name of a OutputDataPoint class of the service The class is in the range of object property:hasOutputDataPoint of a Service.

-

convention	Is short for	replacement rule
&OPERATIONINPUT_1; … &OPERATIONINPUT_n;	Name of a *customAttribute* of the Operation's *<flexContainer>* describing an OperationInput	Shall be identical to the name of a OperationInput class of the Operation The class is in the range of object property:hasInput of an Operation.
&OPERATIONOUTPUT_1; … &OPERATIONOUTPUT_n;	Name of a *customAttribute* of the Operation's *<flexContainer>* describing an OperationOutput	Shall be identical to the name of a OperationOutput class of the Operation. The class is in the range of object property:hasOutput of an Operation.
&SIMPLEDATATYPE;	One out of: xs:NCName , xs:anySimpleType , xs:anyType , xs:anyURI , xs:base64Binary , xs:boolean , xs:decimal , xs:dateTime, xs:double , xs:duration , xs:float , xs:hexBinary , xs:integer , xs:language , xs:nonNegativeInteger , xs:normalizedString , xs:positiveInteger , xs:string , xs:token , xs:unsignedInt , xs:unsignedLong , xs:unsignedShort	Shall be identical to the value of the data property:hasDataType
&RESTRICTIONVALUE;	The value of the restriction	Is a regular expression in case of restriction type &RESTRICTIONTYPE; = xs:pattern, in all other cases a number
&TYPENAME;	Name of the type of a variable (Inut/OutputDataPoint, OperationInput/Output,Thingproperty) or name of a sub-structure in a StructuredTypeVariable	Shall be identical to the name the class of a Variable or sub-structure (i.e. a Variable that is the range of object property:hasSubStructure), adding "Type" to that name. e.g. if the name of an OutputDataPoint class is &OUTPUTDATAPOINT; = "temperature" then the &TYPENAME; is "temperatureType". If "temperature" is a StructuredTypeVariable, containing substructures with &VARIABLENAME_1; = "fahrenheit" and &VARIABLENAME_2; = "accuracy" then the two &TYPENAME;s would be "fahrenheitType" and "accuracyType"
&VARIABLENAME_1; &VARIABLENAME_2; … &VARIABLENAME_n;	name of a sub-structure in a StructuredTypeVariable	Shall be identical to the class name of a Variable that is the range of object property:hasSubStructure

-

convention	Is short for	replacement rule
NOTE 1:	In case the operation produces operation output an instance of the <*flexContainer*> for that operation may have a child-resource of the same <*flexContainer*> specialization type for that Operation to contain the operation output data.	
NOTE 2:	For the kind of restriction (xs:minInclusive, xs:maxInclusive …) of simple data types no convention is given in this table. Its entry in the XSD file is given by the line: <*kind-of-restriction* value="&RESTRICTIONVALUE;"/> where *kind-of-restriction* is given by the value in the range of a specific sub-property of Data Property: hasDataRestriction of a SimpeTypeVariable. It is possible to specify no restrictions, one restriction or multiple restrictions for a simple data type.	
NOTE 3:	Sub-services (range of object property:hasSubService of a Service) are not considered in this release.	

7.2.1.3 Data typing for Variables

7.2.1.3.1 Information on datatypes contained in the ontology

All classes of the Base Ontology that describe data (i.e. InputDataPoint, OutputDataPoint, OperationInput, OperationOutput, ThingProperty are sub-classes of either the class:SimpleTypeVariable or the class StructuredTypeVariable. (This is due to the fact that class:Variable is the disjoint union of classes SimpleTypeVariable and StructuredTypeVariable).

Class:Variable and its sub-classes Class:SimpleTypeVariable and StructuredTypeVariable support (i.e. are the domain class of) properties that allow to specify the datatype of the variable:

- Class:SimpleTypeVariable supports data property:isDataList (range: xsd:Boolean).
  If a class of the ontology that describes the Interworked Device is a sub-class of class:SimpleTypeVariable and "isDataList ***value*** *TRUE*" is specified then the class describes a list data type. Similarly, if "isDataList ***value*** *FALSE*" is specified or property:isDataList is not specified then the class describes a data type describing a single value.

- Class:SimpleTypeVariable supports data property:hasDataType. The range of data property:hasDataType is the set of following strings:

  {"xs:NCName" , "xs:anySimpleType" , "xs:anyType" , "xs:anyURI" , "xs:base64Binary" , "xs:boolean" , "xs:dateTime" , "xs:decimal" , "xs:double" , "xs:duration" , "xs:float" , "xs:hexBinary" , "xs:integer" , "xs:language" , "xs:nonNegativeInteger" , "xs:normalizedString" , "xs:positiveInteger" , "xs:string" , "xs:token" , "xs:unsignedInt" , "xs:unsignedLong" , "xs:unsignedShort"}

This data property indicates that the data type of the Variable is exactly one of these simple XML data types, contained in https://www.w3.org/TR/xmlschema11-2 (see [3] clause 6.3.1).
For example if a class of the ontology that describes the Interworked Device is a sub-class of class:SimpleTypeVariable and " hasDataType ***value*** "xs:integer"" is specified then the class describes an integer value (or, if additionally "isDataList ***value*** TRUE" is specified, then the class describes a list of integer values).

- Class:SimpleTypeVariable also supports data property:hasDataRestriction and its sub-properties:

  - hasDataRestriction_minInclusive,
  - hasDataRestriction_maxInclusive,
  - hasDataRestriction_minExclusive,
  - hasDataRestriction_maxExclusive,
  - hasDataRestriction_length,
  - hasDataRestriction_minLength,
  - hasDataRestriction_maxLength,
  - hasDataRestriction_pattern,

  which specify restrictions on the permissible values of the data.

-

- In addition Class: StructuredTypeVariable supports object property:hasSubStructure (range: class:Variable). This object property allows to create complex, structured data types. Each class in the range of object property:hasSubStructure signifies an element of the structure.

EXAMPLE:

If a class XY of the ontology is a sub-class of class:Variable and supports object property:

- hasSubStructure range class:X (class:X being a sub-class of class:Variable)
- hasSubStructure range class:Y (class:Y being a sub-class of class:Variable)

then class XY is also a sub-class of class: StructuredTypeVariable and describes a structured data type that has data types X and Y as sub structures.

NOTE: The data properties:hasDataType, isDataList and the object property:hasSubStructure are mutually exclusive:
class:SimpleTypeVariable specifies (is defined as a sub-class of:)"hasSubStructure exactly 0 Variable".
A data type cannot at the same time be a simple data type and a structured data type!

7.2.1.3.2 Construction of Simple Data Types

If class:Variable supports:

- data property: isDataList ***value*** FALSE
- data property: hasDataType ***value*** $SIMPLEDATATYPE$

then the XSD for the <*flexContainer*> that contains a *customAttribute* of that type (given by &TYPENAME;) needs to contain the typedefinition for that *customAttribute.*

Table 2: Type definition for Simple Types

```
<!-- ********************** -->
<!-- Ontology based IWK Simple Types -->
<!-- ********************** -->

<xs:simpleType name="&TYPENAME;">
  <xs:restriction base="&SIMPLEDATATYPE;">
    <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
    <xs:minInclusive value="&RESTRICTIONVALUE;"/>
    <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength, xs:pattern -->
    <xs:maxInclusive value="&RESTRICTIONVALUE;"/>
    <!-- end of lines only for restrictions of the values -->
  </xs:restriction>
</xs:simpleType>
```

Example:
if the name of an OutputDataPoint class is &OUTPUTDATAPOINT; = "temperature", which contains a temperature value in Celsius, then a type definition could be

```
<xs:simpleType name="temperatureType">
  <xs:restriction base="xs:float">
    < xs:minInclusive value="−273.15"/>
    < xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
```

-

7.2.1.3.3 List Data Types

If class:Variable supports:

- data property: isDataList ***value*** TRUE
- data property: hasDataType ***value*** $SIMPLEDATATYPE$

then the XSD for the <*flexContainer*> that contains a *customAttribute* of that type (given by &TYPENAME;) needs to contain the typedefinition for that *customAttribute.*

Table 3: Type definition for List Data Types

```
<!--********************** -->
<!-- Ontology based IWK Simple Types (Lists)-->
<!--********************** -->

<xs:simpleType name="&TYPENAME;">
  <xs:restriction>
    <xs:simpleType>
      <xs:list>
        <xs:simpleType>
          <xs:restriction base="&SIMPLEDATATYPE;">
          </xs:restriction>
        </xs:simpleType>
      </xs:list>
    </xs:simpleType>
    <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
    <xs: xs:pattern value="&RESTRICTIONVALUE;"/>
    <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength,
xs:maxLength, xs:pattern -->
    <xs:maxLength value="&RESTRICTIONVALUE;"/>
    <!-- end of lines only for restrictions of the values -->
  </xs:restriction>
</xs:simpleType>
```

Example:
if the name of an OutputDataPoint class is &OUTPUTDATAPOINT; = "temperatureSeries", which contains a list of up to 10 temperature values, then a type definition could be

```
<xs:simpleType name="temperatureSeriesType">
  <xs:restriction>
    <xs:simpleType>
      <xs:list>
        <xs:simpleType>
          <xs:restriction base="xs:float">
          </xs:restriction>
        </xs:simpleType>
      </xs:list>
    </xs:simpleType>
    < xs:minInclusive value="−273.15"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

-

7.2.1.3.4 Structured Data Types

If class:Variable supports:

- data property: isDataList ***value*** FALSE
- object property: hasSubStructure with range class:&VARIABLENAME_1;
- object property: hasSubStructure with range class:&VARIABLENAME_2;
- …
- object property: hasSubStructure with range class:&VARIABLENAME_n;

then the XSD for the <*flexContainer*> that contains that *customAttribute* needs to contain the typedefinition (given by &TYPENAME;) for that *customAttribute*.
Additionally (simple or complex) typedefinitions for the Variables that are in the range of object property: hasSubStructure (given by &TYPENAME_1; … &TYPENAME_n;) need to be contained in the XSD.

Table 4: Type definition for Structured Data Types

```
<!--********************** -->
<!-- Ontology based IWK Complex Types -->
<!--********************** -->

<xs:complexType name="&TYPENAME;">
   <xs:sequence>
      <!-- should include all classes:Variable that are in the range of object property:hasSubStructure  -->
      <xs:element name="&VARIABLENAME_1;"  type=" obi:&TYPENAME_1;"/>
      <!-- ... -->
      <xs:element name="&VARIABLENAME_n;"  type=" obi:&TYPENAME_n;"/>
   </xs:sequence>
</xs:complexType>
```

Example:
if the name of a OutputDataPoint class is &OUTPUTDATAPOINT; = "accurateTemperature" and that "accurateTemperature" is a StructuredTypeVariable, containing substructures with &VARIABLENAME_1; = "fahrenheit" and &VARIABLENAME_2; = "accuracy" then then a type definition could be

```
<xs:complexType name="accurateTemperatureType">
   <xs:sequence>
      <!-- should include all classes:Variable that are in the range of object property:hasSubStructure  -->
      <xs:element name="fahrenheit"  type="fahrenheitType"/>
      <!-- ... -->
      <xs:element name="accuracy"  type="accuracyType"/>
   </xs:sequence>
</xs:complexType>
```

7.2.2 XSD template for sub-classes of Base Ontology class: Device

This clause only applies for the case when an Interworked Device is represented as <*flexContainer*>. In case an Interworked Device is represented as <*AE*> the resource definition of the <*AE*> resource in oneM2M TS-0001 [2] applies.

Specific XSD definition for class:Device to be substituted at the placeholder in the XSD skeleton in clause 7.2.1.2

The value of attribute *containerDefinition* of the specialization of the <*flexContainer*> shall be set as specified in table 1. Example: <xs:element name="containerDefinition" type="//m2m.service.com/IN-CSE-0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines /testDevice.xsd" />.

-

Table 5: Template definition of a *<flexContainer>* for Device

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

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

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

-->

<!DOCTYPE placeholders [    <!--  ==== examples; see description in table 1 ====  -->
<!ENTITY XSDSTORAGE    "ww.XYZ.com_WashingMachines">  <!-- Example targetNamespace value -->
<!ENTITY DEVICE              "testDevice">            <!-- Example name of the device defined in this file-->
<!ENTITY DEVICE_1                  "device_1">          <!--  Example name of sub-device -->
<!ENTITY DEVICE_n                  "device_n">          <!--  Example name of sub-device -->
<!ENTITY SERVICE_1          "service_1">          <!--  Example name of service of this device -->
<!ENTITY SERVICE_n          "service_n">          <!--  Example name of service of this device -->
<!ENTITY THINGPROPERTY_1      "tp_1">          <!--  Example name of thingProperty of this device -->
<!ENTITY THINGPROPERTY_n      "tp_n">          <!--  Example name of thingProperty of this device -->
<!ENTITY TYPENAME_1             "tp_1Type">          <!--  Example type name (of thingProperty) -->
<!ENTITY TYPENAME_2 "variablename_1Type">       <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_3 "variablename_3Type">       <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_n           " tp_nType ">          <!--  Example type name (of thingProperty) -->
<!ENTITY VARIABLENAME_1 "variablename_1"> <!--  Example variable name (of sub-structure) -->
<!ENTITY VARIABLENAME_n "variablename_n"> <!--  Example variable name (of sub-structure) -->
<!ENTITY SIMPLEDATATYPE "xs:integer">              <!--  Example simple datatype -->
<!ENTITY RESTRICTIONVALUE "100">                   <!--  Example value of a type restriction -->
]>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="&XSDSTORAGE;"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="&XSDSTORAGE;"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">

   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>

   <!-- include XSDs of child resources: (sub-)Devices and Services used in this Device -->
   <xs:include schemaLocation="&DEVICE_1;.xsd" />
   <!-- ... -->
   <xs:include schemaLocation="&DEVICE_n;.xsd" />´
   <xs:include schemaLocation="&SERVICE_1;.xsd" />
   <!-- ... -->
   <xs:include schemaLocation="&SERVICE_n;.xsd" />
```

-

```
<xs:element name="&DEVICE;" substitutionGroup="m2m:sg_flexContainerResource">
  <xs:complexType>
    <xs:complexContent>
      <!-- Inherit Common Attributes from data type "flexContainerResource" -->
      <xs:extension base="m2m:flexContainerResource">
        <xs:sequence>

          <!-- Resource Specific Attributes -->

           <!-- nodeLink as custom attribute -->
          <xs:element name="nodeLink" type="xs:anyURI" minOccurs="0"/>
          <!-- all OperationInput- and Outputs are listed here as custom attributes -->
          <xs:element name="&THINGPROPERTY_1;" type=" obi:&TYPENAME_1;" minOccurs="0"/>
          <!-- ... -->
          <xs:element name="&THINGPROPERTY_n;" type=" obi:&TYPENAME_n;" minOccurs="0"/>


          <!-- Child Resources -->

          <xs:choice minOccurs="0" maxOccurs="1">
            <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
            <xs:choice minOccurs="1" maxOccurs="unbounded">

              <!-- Device specific Child Resources (sub-Devices of the Device) -->
              <xs:element ref=" obi:&DEVICE_1;"/>
              <!-- ... -->
              <xs:element ref=" obi:&DEVICE_n;"/>

              <!-- Device specific Child Resources (Services of the Device) -->
              <xs:element ref=" obi:&SERVICE_1;"/>
              <!-- ... -->
              <xs:element ref=" obi:&SERVICE_n;"/>

              <!-- Common Child Resources -->
              <xs:element ref="m2m:semanticDescriptor"/>
              <xs:element ref="m2m:subscription"/>
            </xs:choice>
          </xs:choice>

        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>


<!-- types used for ThingProperties of this Device -->

<!-- ********************** -->
<!-- Ontology based IWK Simple Types -->
<!-- ********************** -->

<xs:simpleType name="&TYPENAME_1;">
  <xs:restriction base="&SIMPLEDATATYPE;"/>
</xs:simpleType>

<xs:simpleType name="&TYPENAME_2;">
  <xs:restriction base="&SIMPLEDATATYPE;">
    <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
    <xs:minInclusive value="&RESTRICTIONVALUE;"/>
    <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
    <xs:maxInclusive value="&RESTRICTIONVALUE;"/>
    <!-- end of lines only for restrictions of the values -->
  </xs:restriction>
</xs:simpleType>
```

-

```
   <!--********************** -->
   <!-- Ontology based IWK Simple Types (Lists)-->
   <!--********************** -->

   <xs:simpleType name="&TYPENAME_3;">
      <xs:restriction>
         <xs:simpleType>
            <xs:list>
               <xs:simpleType>
                  <xs:restriction base="&SIMPLEDATATYPE;">
                  </xs:restriction>
               </xs:simpleType>
            </xs:list>
         </xs:simpleType>
         <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
         <xs: xs:pattern value="&RESTRICTIONVALUE;"/>
         <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
         <xs:maxLength value="&RESTRICTIONVALUE;"/>
         <!-- end of lines only for restrictions of the values -->
      </xs:restriction>
   </xs:simpleType>


   <!--********************** -->
   <!-- Ontology based IWK Complex Types -->
   <!--********************** -->

   <xs:complexType name="&TYPENAME_n;">
      <xs:sequence>
         <!-- should include all classes:Variable that are in the range of object property:hasSubStructure  -->
         <xs:element name="&VARIABLENAME_1;"  type=" obi:&TYPENAME_2;"/>
         <!-- ... -->
         <xs:element name="&VARIABLENAME_n;"  type=" obi:&TYPENAME_3;"/>
      </xs:sequence>
   </xs:complexType>

</xs:schema>
```

7.2.3 XSD template for sub-classes of Base Ontology class: Service

Specific XSD definition for class:Service to be substituted at the placeholder in the XSD skeleton in clause 7.2.1.2.

The value of attribute *containerDefinition* of the specialization of the <*flexContainer*> shall be set as specified in table 1. Example: <xs:element name="containerDefinition" type="//m2m.service.com/IN-CSE-0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines /testService.xsd" />.

Table 6: Template definition of a *<flexContainer>* for Service

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to
understand
```

-

and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations.
No recommendation as to products or vendors is made or should be implied.

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

-->

```
<!DOCTYPE placeholders [    <!--  ==== examples; see description in table 1 ====  -->
<!ENTITY XSDSTORAGE    "ww.XYZ.com_WashingMachines">  <!-- Example targetNamespace value -->
<!ENTITY SERVICE       "testService">          <!-- Example name of the service defined in this file-->
<!ENTITY OPERATION_1   "operation_1">          <!--  Example name of operation -->
<!ENTITY OPERATION_n   "operation_n">          <!--  Example name of operation -->
<!ENTITY INPUTDATAPOINT_1     "idp_1">            <!--  Example name of inputDataPoint of this service -->
<!ENTITY INPUTDATAPOINT_n     "idp_n">            <!--  Example name of inputDataPoint of this service -->
<!ENTITY OUTPUTDATAPOINT_1   "odp_1">      <!--  Example name of outputDataPoint of this service -->
<!ENTITY OUTPUTDATAPOINT_n   "odp_n">      <!--  Example name of outputDataPoint of this service -->
<!ENTITY TYPENAME_1          "idp_1Type">           <!--  Example type name (of DataPoint) -->
<!ENTITY TYPENAME_2 "variablename_1Type">      <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_3 "variablename_3Type">      <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_n          " odp_nType ">          <!--  Example type name (of DataPoint) -->
<!ENTITY VARIABLENAME_1 "variablename_1"> <!--  Example variable name (of sub-structure) -->
<!ENTITY VARIABLENAME_n "variablename_n"> <!--  Example variable name (of sub-structure) -->
<!ENTITY SIMPLEDATATYPE "xs:integer">          <!--  Example simple datatype -->
<!ENTITY RESTRICTIONVALUE "100">               <!--  Example value of a type restriction -->
]>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="&XSDSTORAGE;"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="&XSDSTORAGE;"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">

   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>

   <!-- include XSDs of child resources: Operations used in this Service -->
   <xs:include schemaLocation="&OPERATION_1;.xsd" />
   <!-- ... -->
   <xs:include schemaLocation="&OPERATION_n;.xsd" />


   <xs:element name="&SERVICE;" substitutionGroup="m2m:sg_flexContainerResource">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from data type "flexContainerResource" -->
            <xs:extension base="m2m:flexContainerResource">
               <xs:sequence>

                  <!-- Resource Specific Attributes -->

                  <!-- all Input- and OutputDatapoints are listed here as custom attributes -->
                  <xs:element name="&INPUTDATAPOINT_1;" type=" obi:&TYPENAME_1;" minOccurs="0"/>
                  <xs:element name="&OUTPUTDATAPOINT_1;" type=" obi:&TYPENAME_1;" minOccurs="0"/>
                  <!-- ... -->
                  <xs:element name="&INPUTDATAPOINT_n;" type=" obi:&TYPENAME_n;" minOccurs="0"/>
                  <xs:element name="&OUTPUTDATAPOINT_n;" type=" obi:&TYPENAME_n;" minOccurs="0"/>
```

-

```
                <!-- Child Resources -->

                <xs:choice minOccurs="0" maxOccurs="1">
                  <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                    <!-- Service specific Child Resources (Operations of the Service) -->
                    <xs:element ref=" obi:&OPERATION_1;"/>
                    <!-- ... -->
                    <xs:element ref=" obi:&OPERATION_n;"/>

                    <!-- Common Child Resources -->
                    <xs:element ref="m2m:semanticDescriptor"/>
                    <xs:element ref="m2m:subscription"/>
                  </xs:choice>
                </xs:choice>

              </xs:sequence>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
    </xs:element>


<!-- types used for Datapoints of this Service -->

    <!-- ********************** -->
    <!-- Ontology based IWK Simple Types -->
    <!-- ********************** -->

    <xs:simpleType name="&TYPENAME_1;">
      <xs:restriction base="&SIMPLEDATATYPE;"/>
    </xs:simpleType>

    <xs:simpleType name="&TYPENAME_2;">
      <xs:restriction base="&SIMPLEDATATYPE;">
        <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
        <xs:minInclusive value="&RESTRICTIONVALUE;"/>
        <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
        <xs:maxInclusive value="&RESTRICTIONVALUE;"/>
        <!-- end of lines only for restrictions of the values -->
      </xs:restriction>
    </xs:simpleType>


    <!--********************** -->
    <!-- Ontology based IWK Simple Types (Lists)-->
    <!--********************** -->

    <xs:simpleType name="&TYPENAME_3;">
      <xs:restriction>
        <xs:simpleType>
          <xs:list>
            <xs:simpleType>
              <xs:restriction base="&SIMPLEDATATYPE;">
              </xs:restriction>
            </xs:simpleType>
          </xs:list>
        </xs:simpleType>
        <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
        <xs:pattern value="&RESTRICTIONVALUE;"/>
        <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
        <xs:maxLength value="&RESTRICTIONVALUE;"/>
        <!-- end of lines only for restrictions of the values -->
      </xs:restriction>
    </xs:simpleType>
```

-

```
  <!--********************** -->
  <!-- Ontology based IWK Complex Types -->
  <!--********************** -->

  <xs:complexType name="&TYPENAME_n;">
     <xs:sequence>
        <!-- should include all classes:Variable that are in the range of object property:hasSubStructure  -->
        <xs:element name="&VARIABLENAME_1;"  type=" obi:&TYPENAME_2;"/>
        <!-- ... -->
        <xs:element name="&VARIABLENAME_n;"  type=" obi:&TYPENAME_3;"/>
     </xs:sequence>
  </xs:complexType>



</xs:schema>
```

7.2.4 XSD template for sub-classes of Base Ontology class: Operation

Specific XSD definition for class:Operation to be substituted at the placeholder in the XSD skeleton in clause 7.2.1.2.

The value of attribute *containerDefinition* of the specialization of the <*flexContainer*> shall be set as specified in table 1. Example: <xs:element name="containerDefinition" type="//m2m.service.com/IN-CSE-0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines /testOperation.xsd" />.

Table 7: Template definition of a *<flexContainer>* for Operation

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

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

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

-->

<!DOCTYPE placeholders [    <!--  ==== examples; see description in table 1 ====  -->
<!ENTITY XSDSTORAGE    "ww.XYZ.com_WashingMachines">  <!-- Example targetNamespace value -->
<!ENTITY OPERATION     "testOperation">               <!--  Example name of operation defined in this file -->
<!ENTITY OPERATIONINPUT_1    "oip_1">                 <!--  Example name of operationInput of this service -->
```

-

```
<!ENTITY OPERATIONINPUT_n    "oip_n">              <!--  Example name of operationInput of this service -->
<!ENTITY OPERATIONOUTPUT_1 "oop_1">              <!--  Example name of operationOutput of this service -->
<!ENTITY OPERATIONOUTPUT_n "oop_n">              <!--  Example name of operationOutput of this service -->
<!ENTITY TYPENAME_1             "oip_1Type">          <!--  Example type name (of operationInput) -->
<!ENTITY TYPENAME_2 "variablename_1Type">       <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_3 "variablename_3Type">       <!--  Example type name (of sub-structure) -->
<!ENTITY TYPENAME_n           " oop_nType ">          <!--  Example type name (of operationOutput) -->
<!ENTITY VARIABLENAME_1 "variablename_1"> <!--  Example variable name (of sub-structure) -->
<!ENTITY VARIABLENAME_n "variablename_n"> <!--  Example variable name (of sub-structure) -->
<!ENTITY SIMPLEDATATYPE "xs:integer">            <!--  Example simple datatype -->
<!ENTITY RESTRICTIONVALUE "100">                 <!--  Example value of a type restriction -->
]>

<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="&XSDSTORAGE;"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="&XSDSTORAGE;"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">

   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>

   <!--  for operations no includes needed from obi: namespace -->

   <xs:element name="&OPERATION;" substitutionGroup="m2m:sg_flexContainerResource">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from data type "flexContainerResource" -->
            <xs:extension base="m2m:flexContainerResource">
               <xs:sequence>

                  <!-- Resource Specific Attributes for operations-->

                  <!-- all OperationInput- and Outputs are listed here as custom attributes -->
                  <xs:element name="&OPERATIONINPUT_1;" type=" obi:&TYPENAME_1;" minOccurs="0"/>
                  <xs:element name="&OPERATIONOUTPUT_1;" type=" obi:&TYPENAME_n;" minOccurs="0"/>
                  <!-- ... -->
                  <xs:element name="&OPERATIONINPUT_n;" type=" obi:&TYPENAME_1;" minOccurs="0"/>
                  <xs:element name="&OPERATIONOUTPUT_n;" type=" obi:&TYPENAME_n;" minOccurs="0"/>


                  <!-- Child Resources -->

                  <xs:choice minOccurs="0" maxOccurs="1">
                     <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1"
maxOccurs="unbounded"/>
                     <xs:choice minOccurs="1" maxOccurs="unbounded">

                        <!-- Child Resource for Operations that have OperationOutputs is an Operation of the same type -->
                        <xs:element ref="&OPERATION;"/>

                        <!-- Common Child Resources -->
                        <xs:element ref="m2m:semanticDescriptor"/>
                        <xs:element ref="m2m:subscription"/>
                     </xs:choice>
                  </xs:choice>

               </xs:sequence>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>


   <!-- types used for OperationInputs and -Outputs of this Operation -->

   <!-- ********************** -->
   <!-- Ontology based IWK Simple Types -->
```

-

```
    <!-- ********************** -->

    <xs:simpleType name="&TYPENAME_1;">
       <xs:restriction base="&SIMPLEDATATYPE;"/>
    </xs:simpleType>

    <xs:simpleType name="&TYPENAME_2;">
       <xs:restriction base="&SIMPLEDATATYPE;">
          <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restriction types possible -->
          <xs:minInclusive value="&RESTRICTIONVALUE;"/>
          <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
          <xs:maxInclusive value="&RESTRICTIONVALUE;"/>
          <!-- end of lines only for restrictions of the values -->
       </xs:restriction>
    </xs:simpleType>


    <!--********************** -->
    <!-- Ontology based IWK Simple Types (Lists)-->
    <!--********************** -->

    <xs:simpleType name="&TYPENAME_3;">
       <xs:restriction>
          <xs:simpleType>
             <xs:list>
                <xs:simpleType>
                   <xs:restriction base="&SIMPLEDATATYPE;">
                   </xs:restriction>
                </xs:simpleType>
             </xs:list>
          </xs:simpleType>
          <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
          <xs:pattern value="&RESTRICTIONVALUE;"/>
          <!-- ... xs:minInclusive, xs:maxInclusive, xs:minExclusive, xs:maxExclusive, xs:length, xs:minLength, xs:maxLength,
xs:pattern -->
          <xs:maxLength value="&RESTRICTIONVALUE;"/>
          <!-- end of lines only for restrictions of the values -->
       </xs:restriction>
    </xs:simpleType>


    <!--********************** -->
    <!-- Ontology based IWK Complex Types -->
    <!--********************** -->

    <xs:complexType name="&TYPENAME_n;">
       <xs:sequence>
          <!-- should include all classes:Variable that are in the range of object property:hasSubStructure  -->
          <xs:element name="&VARIABLENAME_1;"  type=" obi:&TYPENAME_2;"/>
          <!-- ... -->
          <xs:element name="&VARIABLENAME_n;"  type=" obi:&TYPENAME_3;"/>
       </xs:sequence>
    </xs:complexType>



</xs:schema>
```

-

Annex A (informative): Example for ontology based interworking

A.1 Overview

This example extends the example - a (simplified) washing machine - of clause B.1.3.3 of oneM2M TS-0012 [3]. However in contrast to oneM2M TS-0012 in this example it is assumed that the washing machine is not a 'native' oneM2M application but is interworked with the oneM2M System via ontology based interworking.

In contrast to oneM2M TS-0012 the semantic description (contained in the *descriptor* attribute of the <*semanticDescriptor*> resource) is not of interest but rather the structure of the specialized <*flexContainer*> resources to which these <*semanticDescriptor*>s belong.
Also Functions and Commands (which are only described in RDF form in the sematic descriptors) are not considered, but only Services, Input- / OutputDatapoints and Operations, which are the resources containing data for communication with the washing machine are handled here.

The example given here is a (simplified) washing machine, based on class:WashingMachine, specified in SAREF [i.2].

It is assumed the washing machine has been manufactured by manufacturer XYZ.
The ontology that describes XYZ washingmachines is identified by the IRI: http://www.XYZ.com/WashingMachines.

It is assumed the XYZ washing machines are compliant with SAREF. Also, as described in clause B.1 of oneM2M TS-0012 [3], SAREF is compliant with the oneM2M base ontology.
That implies that every class/property of the XYZ ontology is either a subclass/sub-property of SAREF or - if no suitable class/property exists in SAREF - is a subclass/sub-property of the base ontology.

The classes/properties that are taken from SAREF are prefixed with: "saref:"
The classes/properties that are taken from XYZ ontology are prefixed with: "XYZ:"
The classes/properties that are taken from the base ontology are prefixed with: "BO:"

- The model of the type of washing machine is class XYZ:**XYZ_Cool**
  - which is a sub-class of class saref:WashingMachine
    - which in turn is subclass of BO:InterworkedDevice
- The state of the washing machine is given by XYZ:**WashingMachineStatus** that can take the values "WASHING" or "STOPPED" or "ERROR".
  - XYZ:WashingMachineStatus is a sub-class of saref:state
    - which in turn is subclass of BO:OutputDataPoint
- This WashingMachineStatus is provided as an the OutputDataPoint of a service XYZ:**MonitorService** which exposes the MonitoringFunction to the network.
  - Class XYZ:MonitorService is a sub-class of saref:Service
    - which in turn is subclass of BO:Service
- The related function of the washing machine is described by class XYZ:MonitoringFunction.
  - Class XYZ:MonitoringFunction is a sub-class of saref: Function
    - which in turn is subclass of BO:Function
- The function of the washing machine to control the washing machine is in class XYZ:StartStopFunction which has three commands: ON_Command, OFF_Command, Toggle_Command.

-

NOTE: Since classes for Functions and Commands are not needed for interworking as they only appear in the RDF description in <sematicDescriptor> resources the XYZ:MonitoringFunction and XYZ:StartStopFunction are not further considered here. They are, however, described in the related example of clause B.1.3.3 of oneM2M TS-0012 [3].

- The related service of the washing machine that represents that XYZ:StartStopFunction is class: XYZ:**SwitchOnService.** It has:
  an InputDataPoint: XYZ:**BinaryInput** (to expose command ON_Command and OFF_Command) and
  an Operation: XYZ:**ToggleBinary** (to expose command Toggle_Command)
  - Class XYZ: SwitchOnService is a sub-class of saref:SwitchOnService
    - which in turn is subclass of BO:Service
- InputDataPoint: XYZ:BinaryInput is a sub-class of BO:InputDataPoint
- Operation: XYZ:**ToggleBinary** is a sub-class of BO:Operation
- XYZ describes this type of washing machine with a XYZ:**Description** as "Very cool Washing Machine"
  - XYZ:Description os a sub-class of BO:ThingProperty

The following figure shows the relationship of the ontology that describes model XYZ_cool of XYZ washingmachines with the oneM2M base ontology.

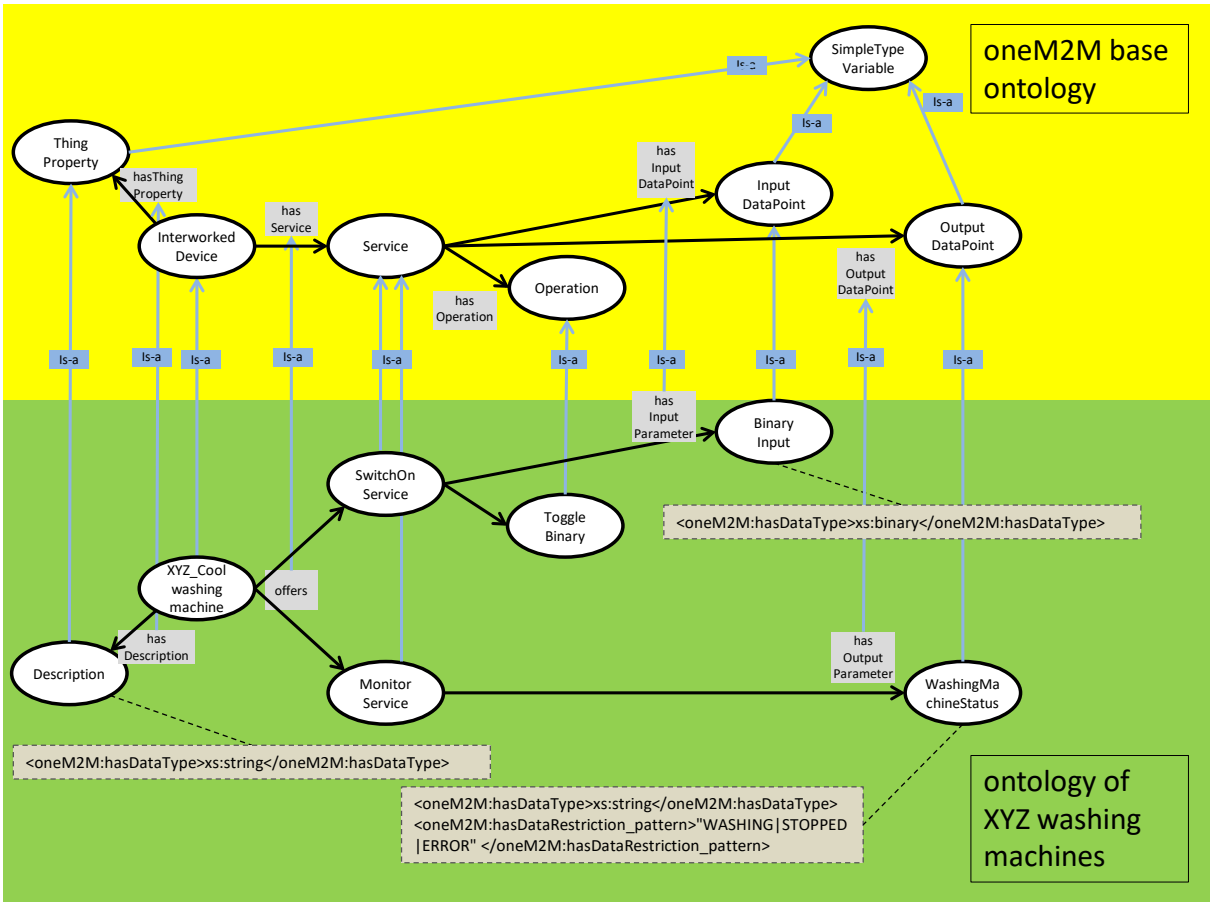

Figure 7: Relationship of the ontology that describes model XYZ_cool of XYZ washingmachines with the oneM2M base ontology

-

A.2 XSDs created by the IPE

A.2.1 XSD storage <*container*>

According to clause 7.2.1.1 the IPE creates a <*container*> resource that acts as a XSD-storage and that is a child resource of the IPE's <*AE*>.

Derived from the IRI of the ontology: http://www.XYZ.com/WashingMachines, following the rule given in table 1 of clause 7.2.1.2 the resource name for that <*container*> resource (in the table the placeholder is called &XSDSTORAGE;) will be www.XYZ.com_WashingMachines.

The IPE will also create <*contentInstance*> resources as child-resources of the XSD-storage <*container*> resource with the following resource names (see &XSDFILE; in table 1 of 7.2.1.2):

- XYZ_Cool.xsd … contains the XSD for the <*flexContainer*> resource of the Interworked Device type XYZ_Cool
- SwitchOnService.xsd … contains the XSD for the <*flexContainer*> resource of the Service type SwitchOnService
- MonitorService.xsd … contains the XSD for the <*flexContainer*> resource of the Service type MonitorService
- ToggleBinary.xsd … contains the XSD for the <*flexContainer*> resource of the Operation type ToggleBinary

When the IPE crates a new <*flexContainer*> specialization of these types then the attribute *containerDefinition* will contain the absolute, hierarchical address of the <*contentinstance*> that contains the related XSD.
E.g. for the SwitchOnService it could have the value:
//m2m.service.com/IN-CSE0001/bigFatCse/name_of_IPE_AE/ww.XYZ.com_WashingMachines/SwitchOnService.xsd

A.2.2 XSD for the Interworked Device type XYZ_Cool

Following the principles of clause 7.2.2 an Interworked Device is represented as <*flexContainer*>.

According to the the convention in table 1 of clause 7.2.1.2 the placeholders and their substitutions that are needed for a <*flexContainer*>, representing an Interworked Device are:

• &XSDSTORAGE;	is substituted by	www.XYZ.com_WashingMachines
• &DEVICE;	is substituted by	XYZ_Cool
• &SERVICE_1;	is substituted by	SwitchOnService
• &SERVICE_2;	is substituted by	MonitorService
• THINGPROPERTY_1;	is substituted by	Description
• &TYPENAME_1;	is substituted by	DescriptionType
• &SIMPLEDATATYPE;	is substituted by	xs:string

The XSD in the <*contentInstance*> with resource name XYZ_Cool.xsd will contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
```

-

-->

```
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="www.XYZ.com_WashingMachines"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="www.XYZ.com_WashingMachines"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">


   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>


   <!-- include XSDs of child resources: (sub-)Devices and Services used in this Device -->
   <xs:include schemaLocation="SwitchOnService.xsd" />
   <xs:include schemaLocation="MonitorService.xsd" />


   <xs:element name="XYZ_Cool" substitutionGroup="m2m:sg_flexContainerResource">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from data type "flexContainerResource" -->
            <xs:extension base="m2m:flexContainerResource">
               <xs:sequence>

                  <!-- Resource Specific Attributes -->

                  <!-- nodeLink as custom attribute -->
                  <xs:element name="nodeLink" type="xs:anyURI" minOccurs="0"/>
                  <!-- all OperationInput- and Outputs are listed here as custom attributes -->
                  <xs:element name="Description" type=" obi:DescriptionType" minOccurs="0"/>

                  <!-- Child Resources -->

                  <xs:choice minOccurs="0" maxOccurs="1">
                     <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
                     <xs:choice minOccurs="1" maxOccurs="unbounded">

                        <!-- Device specific Child Resources (sub-Devices of the Device) -->

                        <!-- Device specific Child Resources (Services of the Device) -->
                        <xs:element ref=" obi:SwitchOnService"/>
                        <xs:element ref=" obi:MonitorService"/>

                        <!-- Common Child Resources -->
```

-

```
                    </xs:choice>
                </xs:choice>

            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>


<!-- types used for ThingProperties of this Device -->

<!-- ********************** -->
<!-- Ontology based IWK Simple Types -->
<!-- ********************** -->

<xs:simpleType name="DescriptionType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>

</xs:schema>
```

A.2.3 XSD for the Service type SwitchOnService

Following the principles of clause 7.2.3 a Service is represented as <*flexContainer*>.

According to the the convention in table 1 of clause 7.2.1.2 the placeholders and their substitutions that are needed for a <*flexContainer*>, representing an Interworked Device are:

• &XSDSTORAGE;	is substituted by	www.XYZ.com_WashingMachines
• &SERVICE;	is substituted by	SwitchOnService
• &OPERATION_1;	is substituted by	ToggleBinary
• &INPUTDATAPOINT_1;	is substituted by	BinaryInput
• &TYPENAME_1;	is substituted by	ToggleBinaryType
• &SIMPLEDATATYPE;	is substituted by	xs:boolean

The XSD in the <*contentInstance*> with resource name SwitchOnService.xsd will contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

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

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR
SUFFICIENT OR CONFORMS TO ANY STATUTE,
GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF
```

-

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

-->
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="www.XYZ.com_WashingMachines"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="www.XYZ.com_WashingMachines"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">

   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>



   <!-- include XSDs of child resources: Operations used in this Service -->
   <xs:include schemaLocation="ToggleBinary.xsd" />


   <xs:element name="SwitchOnService" substitutionGroup="m2m:sg_flexContainerResource">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from data type "flexContainerResource" -->
            <xs:extension base="m2m:flexContainerResource">
               <xs:sequence>

                  <!-- Resource Specific Attributes -->

                  <!-- all Input- and OutputDatapoints are listed here as custom attributes -->
                  <xs:element name="BinaryInput" type=" obi:BinaryInputType" minOccurs="0"/>


                  <!-- Child Resources -->

                  <xs:choice minOccurs="0" maxOccurs="1">
                     <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
                     <xs:choice minOccurs="1" maxOccurs="unbounded">

                        <!-- Service specific Child Resources (Operations of the Service) -->
                        <xs:element ref=" obi:ToggleBinary"/>

                        <!-- Common Child Resources -->
                        <xs:element ref="m2m:semanticDescriptor"/>
                        <xs:element ref="m2m:subscription"/>
                     </xs:choice>
                  </xs:choice>

               </xs:sequence>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>


   <!-- types used for Datapoints of this Service -->

   <!-- ********************** -->
   <!-- Ontology based IWK Simple Types -->
   <!-- ********************** -->

   <xs:simpleType name=" BinaryInputType ">
      <xs:restriction base="xs:boolean"/>
```

-

```
    </xs:simpleType>


</xs:schema>
```

A.2.4 XSD for the Service type MonitorService

Following the principles of clause 7.2.3 a Service is represented as <*flexContainer*>.

According to the the convention in table 1 of clause 7.2.1.2 the placeholders and their substitutions that are needed for a <*flexContainer*>, representing an Interworked Device are:

• &XSDSTORAGE;	is substituted by	www.XYZ.com_WashingMachines
• &SERVICE;	is substituted by	MonitorService
• &OUTPUTDATAPOINT_1;	is substituted by	WashingMachineStatus
• &TYPENAME_1;	is substituted by	WashingMachineStatusType
• &SIMPLEDATATYPE;	is substituted by	xs:string
• &RESTRICTIONVALUE;	is substituted by	"WASHING"|"STOPPED"|"ERROR"

The XSD in the <*contentInstance*> with resource name MonitorService.xsd will contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

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

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

-->
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="www.XYZ.com_WashingMachines"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="www.XYZ.com_WashingMachines"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">
```

-

```
<xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
<xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>



<xs:element name="MonitorService" substitutionGroup="m2m:sg_flexContainerResource">
   <xs:complexType>
      <xs:complexContent>
         <!-- Inherit Common Attributes from data type "flexContainerResource" -->
         <xs:extension base="m2m:flexContainerResource">
            <xs:sequence>

               <!-- Resource Specific Attributes -->

               <!-- all Input- and OutputDatapoints are listed here as custom attributes -->
               <xs:element name="WashingMachineStatus" type="obi:WashingMachineStatusType" minOccurs="0"/>


               <!-- Child Resources -->

               <xs:choice minOccurs="0" maxOccurs="1">
                  <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                     <!-- Service specific Child Resources (Operations of the Service) -->

                     <!-- Common Child Resources -->

                     <xs:element ref="m2m:semanticDescriptor"/>
                     <xs:element ref="m2m:subscription"/>

                  </xs:choice>
               </xs:choice>

            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>


<!-- types used for Datapoints of this Service -->

<!-- ********************** -->
<!-- Ontology based IWK Simple Types -->
<!-- ********************** -->

<xs:simpleType name="WashingMachineStatusType">
   <xs:restriction base="xs:string">
   <!-- next lines for restrictions of the values. Remove not needed ones, Multiple restrictiontypes possible -->
      <xs:pattern value= "WASHING|STOPPED|ERROR"/>
   </xs:restriction>
</xs:simpleType>


</xs:schema>
```

-

A.2.5 XSD for the Operation type ToggleBinary

Following the principles of clause 7.2.3 a Service is represented as <*flexContainer*>.

According to the the convention in table 1 of clause 7.2.1.2 the placeholders and their substitutions that are needed for a <*flexContainer*>, representing an Interworked Device are:

• &XSDSTORAGE;	is substituted by	www.XYZ.com_WashingMachines
• &OPERATION;	is substituted by	ToggleBinary

The XSD in the <*contentInstance*> with resource name ToggleBinary.xsd will contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright Notification

The oneM2M Partners authorize you to copy this document, provided that you retain all copyright and other proprietary notices
contained in the original materials on any copies of the materials and that you comply strictly with these terms.
This copyright permission does not constitute an endorsement of the products or services, nor does it encompass the granting of
any patent rights. The oneM2M Partners assume no responsibility for errors or omissions in this document.
© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC). All rights reserved.

Notice of Disclaimer & Limitation of Liability

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

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

-->

<xs:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="www.XYZ.com_WashingMachines"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m2m="http://www.onem2m.org/xml/protocols"
   xmlns:obi="www.XYZ.com_WashingMachines"
   elementFormDefault="unqualified" attributeFormDefault="unqualified">


   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-semanticDescriptor-v3_0_0.xsd"/>
   <xs:import namespace="http://www.onem2m.org/xml/protocols" schemaLocation="CDT-subscription-v3_0_0.xsd"/>


   <!--  for operations no includes needed from obi: namespace -->

   <xs:element name="ToggleBinary" substitutionGroup="m2m:sg_flexContainerResource">
      <xs:complexType>
         <xs:complexContent>
            <!-- Inherit Common Attributes from data type "flexContainerResource" -->
            <xs:extension base="m2m:flexContainerResource">
               <xs:sequence>
```

-

```
                <!-- Resource Specific Attributes for operations-->

                <!-- all OperationInput- and Outputs are listed here as custom attributes -->


                <!-- Child Resources -->

                <xs:choice minOccurs="0" maxOccurs="1">
                   <xs:element name="childResource" type="m2m:childResourceRef" minOccurs="1" maxOccurs="unbounded"/>
                   <xs:choice minOccurs="1" maxOccurs="unbounded">

                      <!-- Child Resource for Operations that have OperationOutputs is an Operation of the same type -->

                      <!-- Common Child Resources -->
                      <xs:element ref="m2m:semanticDescriptor"/>
                      <xs:element ref="m2m:subscription"/>
                   </xs:choice>
                </xs:choice>

             </xs:sequence>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
 </xs:element>


 <!-- types used for OperationInputs and -Outputs of this Operation -->


</xs:schema>
```

-

History

Publication history		
V3.0.2	April 2018	Release 3 - Publication