ONEM2M TECHNICAL REPORT

Document Number	TR-0017-V2.0.0
Document Name:	Home Domain Abstract Information Model
Date:	2016-August-30
Abstract:	This technical report defines an abstract information model for the home domain uses cases. Also, this technical report intends to define how the developed abstract information model for a device could be represented in the CSEs of the oneM2M system.

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

Notice of Disclaimer & Limitation of Liability

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

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

Contents

1 Scope

The present document allows application developers to describe the status of devices as resources on oneM2M-based platform in various ways. Thus different application developers can create different resource trees even when they build the same kinds of applications. Moreover when handling the same kinds of devices from different vendors on M2M platforms, application developers may create disunited resource trees without common information model.

In order to solve such issues, the present document intends to provide the common and unified APIs on one M2M platform for the home domain by defining an abstract information model for the home domain devices such as TV, refrigerator, air conditioner, smart meter, and lighting equipment. The definition of the abstract information model will be based on data models that currently exist in the home domain. The home domain abstract information model does not intend to define the interworking function between the oneM2M system and protocols of the data models from which the abstract data model is defined.

Also, the present document intends to define how the developed abstract information model for a device could be represented in the CSEs of the oneM2M system.

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 reference document (including any amendments) applies.

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

Not applicable.

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 reference 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] oneM2M TS-0011: "Definitions and Acronyms".

[i.3] AllJoyn System Description version 14.06, September 26th, 2014.

[i.4] Home Appliances & Entertainment (HAE) Service Framework Project.

NOTE: Available at https://wiki.allseenalliance.org/hae.

[i.5] HomeKit Developer Site.

NOTE: Available at https://developer.apple.com/homekit/.

[i.6] Designing Accessories for iOS and OS X, WWDC14.

NOTE: Available at https://developer.apple.com/videos/wwdc/2014/.

[i.7] oneM2M TS-0004: "Service Layer Core Protocol Specification".

[i.8] ECHONET Specification Ver.2.11.

NOTE: Available at http://echonet.jp/spec_v211_en/.

[i.9] ECHONET Lite Specification Ver.1.11.

NOTE: Available at http://echonet.jp/spec_v111_lite_en/.

[i.10] ECHONET APPENDIX, Detailed Requirements for ECHONET Device Objects.

NOTE: Available at http://echonet.jp/spec_object_rf_en/.

[i.11] IEC 62394: "Service diagnostic interface for consumer electronics products and networks - Implementation for echonet".

[i.12] ISO/IEC 24767-1: "Information technology -- Home network security -- Part 1: Security requirements".

[i.13] ISO/IEC 24767-2: "Information technology -- Home network security -- Part 2: Internal security services: Secure Communication Protocol for Middleware (SCPM)".

[i.14] IEC 62480: "Multimedia home network - Network interfaces for network adapter".

[i.15] ISO/IEC 14543-4-1: "Information technology -- Home electronic system (HES) architecture -- Part 4-1: Communication layers -- Application layer for network enhanced control devices of HES Class 1".

[i.16] ISO/IEC 14543-4-2: "Information technology -- Home electronic system (HES) architecture -- Part 4-2: Communication layers -- Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1".

[i.17] IEC 62457: "Multimedia home networks - Home network communication protocol over IP for multimedia household appliances".

[i.18] oneM2M TS-0023: "Authorization Architecture for Supporting Heterogeneous Access Control Policies".

[i.19] oneM2M TS-0001: "Functional Architecture".

3 Abbreviations

For the purposes of the present document, the abbreviations given in oneM2M TS-0011 [i.19] and the following apply:

IoE Internet of Everything

4 Conventions

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

5 Home Domain Abstract Information Model

5.1 Anaysis of Abstract Information Models

5.1.1 AllJoyn (AllSeen Alliance)

The AllJoyn system is a proximity-based, peer-to-peer communication platform that provides a framework for enabling communication among internet of everything (IoE) devices across heterogeneous distributed systems [i.3]. Especially, there is a specific project to develop the common way of controlling and monitoring Home Appliances & Entertainment (HAE) devices (such as airconditioner, refrigerator, washer), regardless of device manufacturers [i.4]. In order to achieve this issue, HAE service frame project specifies standard AllJoyn interfaces for controlling and monitoring HAE devices as shown in figure 5.1.1-1.

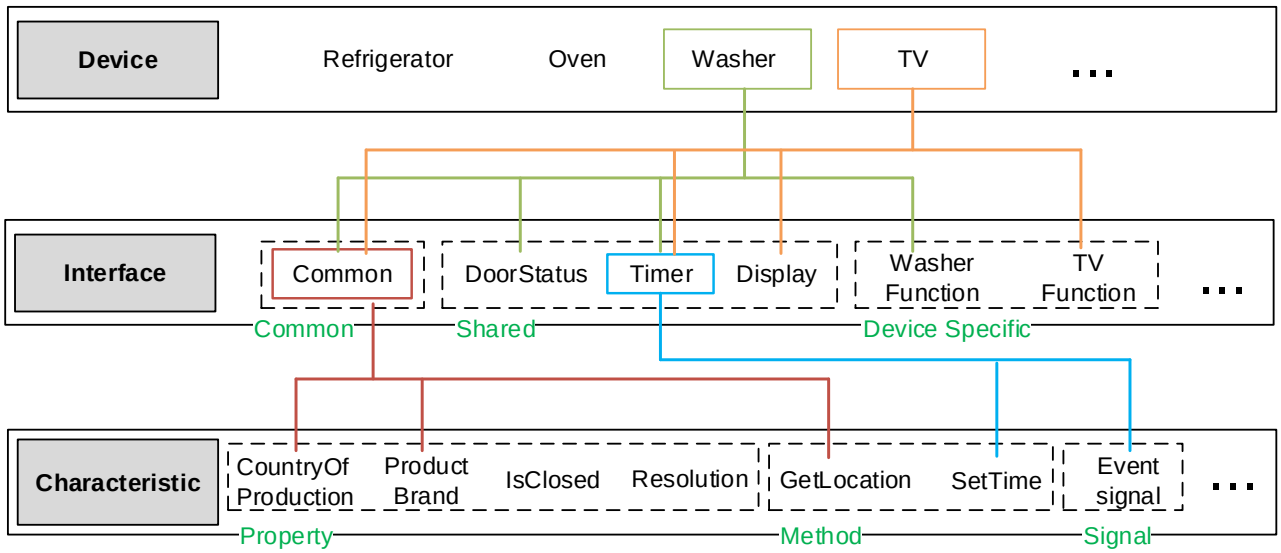

Figure 5.1.1-1: Information Model in AllJoyn

In the AllJoyn information model, 'Device' which represents physical home appliance (such as refrigerator, oven, washer, TV) should have 'Interface(s)' that consist of 'Characteristic(s)'.

'Characteristic' specifies pre-defined device properties(such as country of production, product brand, status of door, display resolution), method(such as getting location, setting the timer) and signal(similar to notification). By combining one or more pre-defined characteristic(s), a specific 'Interface' which provides a particular service(such as subscription or control of the device) can be defined. Furthermore, the AllJoyn system categorizes the 'Interface' into common interface, shared interface and device specific interface for more efficient management. Although the complexity is increased as arraging into the service set, this approach enhances the resource efficiency by resusing the 'Interface'.

5.1.2 HomeKit (Apple Inc.)

HomeKit is a framework in iOS for communicating with and controlling connected accessories in a home domain environment [i.5]. Among developments related document, HomeKit Accessory Profiles defines the information model for some services based on the 'Charateristic' and 'Service' [i.6].

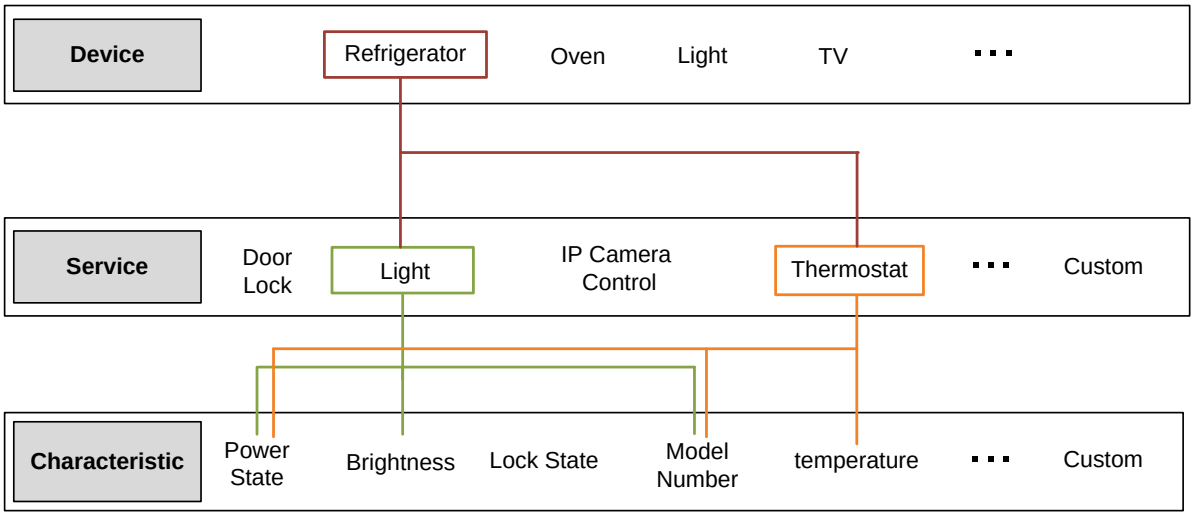

Figure 5.1.2-1: Information Model in HomeKit

As shown in figure 5.1.2-1, the chracteristic specifies pre-defined characteristics and their properties (such as power state, brightness, lock state, model number). Based on the chracteristic, a specific service (such as door lock, light, thermostat) can be defined one or more chracteristic(s). Especially, the 'Custom' value is considered for the un-predefined characteristic and service. As a result, a specific device may have one or more service set(s). Compared to the AllJoyn system approach, the characteristic set in the Homekit are equivalent to the characteristic in the AllJoyn. The HomeKit system has moduled service sets by using characteristic set(s). It is a very similar approach with the 'Interface' in the AllJoyn system.

5.1.3 SmartHome Device Template (HGI)

5.1.3.1 Introduction

The SDT (SmartHome Device Template) is an initiative from HGI to find consensus amongst various SDOs and industry alliances to derive a common approach for device modelling. HGI and partners have the approach to agree on a set of automation commands following a common syntax which are sufficient to model most home appliance functions. The key goals of the SDT are:

1) keep it simple, especially for manufacturers to contribute device information;

2) modularity for functions and device types;

3) make it easy for developers to create unified APIs;

4) be independent of underlying home-area network technologies;

5) enable extendibility of the system in place without service interruption;

6) allow a pass-through mechanism to enable use of proprietary or technology-specific functions.

The SDT approach is to define re-usable basic functions (or services) (labelled "ModuleClass" in figure 5.1.3.1-1) which can represent the typical functions found in many home automation systems, such as "on/off", "dim a lamp", "receive events from binary sensor", "read data from sensor", etc. Each ModuleClass is composed of a (small) number of actions, datapoint read/write operations, or asynchronous events. For example, an "on/off" ModuleClass would consist perhaps of just one Action, but a "ReadKeypad" Action might have a number of possible events, each with some data value and (usually) a sequence-ID or timestamp start/stop to indicate when and how long each key was pressed.

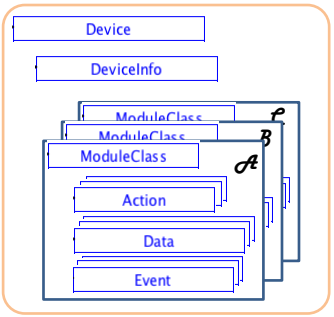

Figure 5.1.3.1-1: SmartHome Device Template (XSD) for a generic device (simplification of draft, under discussion with SDOs)

The SDT represents the device models introduced in figure 5.1.3.1-1 by using an XSD schema to allow formal checking of compliance for XML device descriptions of specific appliances. The modularity goal in the XSD schema is achieved with **re-usable XSD fragments** ("ModuleClass" in figure 5.1.3.1-1).

Complex devices or appliances can then be described by an appropriate set or collection of the agreed XSD fragments (ModuleClasses), as indicated in figure 5.1.3.1-1, which also shows an optional DeviceInfo XSD fragment to allow recording of static information such as device manufacturer name, device firmware version, etc.

Note that the SDT concept of ModuleClasses is similar to the HomeKit concept of "services".

HGI has discussed with many SDOs to validate the concept. Consultation with various industry fora continues, to determine an appropriate set of commonly used ModuleClasses, which however allows extensions. SDT is designed to take into account the list of "services" compiled by the SAREF project.

The SDT supports the use of a set of templates for generic devices or appliances (e.g. for a dimmable lamp, a basic washing machine, etc, which would be specific instances of the "Device" object shown in figure 5.1.3.1-2) which form the basis of APIs used by application developers. These templates can also be referenced by manufacturers creating XML documents to describe their specific products. For example, the SDT enables specification of a generic washing machine template, with on/off, set-wash-temperature, pause and a few other commands, which could be referenced by a manufacturer as the schema for a XML description of a basic model washing machine. The SDT allows for vendor-specific additional commands (ModuleClasses) to suit specific product types.

An example of how three different generic devices/appliances might be modelled using 4 different ModuleClasses is shown in figure 5.1.3.1-2. Data values (DataPoints) which might need to be read/written during operation of the devices are shown as the lowest grouping in the figure (DataPoints/Characteristics).

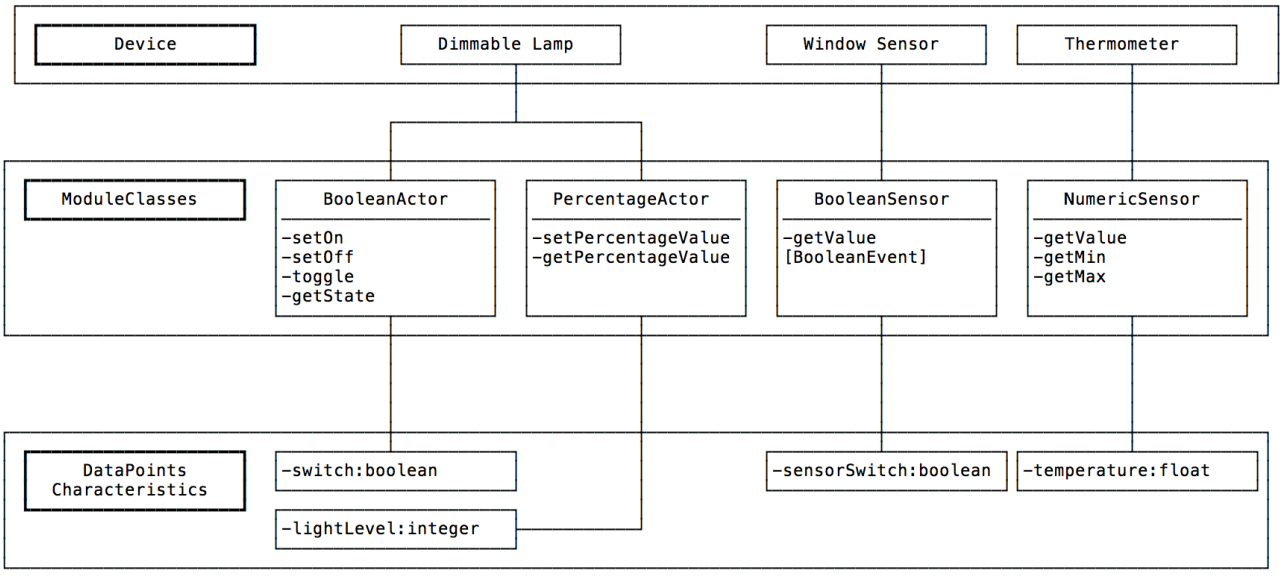

Figure 5.1.3.1-2: SmartHome Template for 3 examples of generic devices

Figure 5.1.3.4-1 showing the SDT structure in more detail is shown in clause 5.1.3.4. It is sufficiently flexible to allow representation of e.g. the SAREF ontology or the future OneM2M ontology.

5.1.3.2 Definitions

This clause provides an overview about the SDT 3.0 definitions and element hierarchy. Terms to be described in detail in this clause are:

Table 5.1.3.2-1: Definitions of SDT Elements

Term	Definition
Domain	Unique name, or "wrapper" which acts like a namespace, set by the organization creating the SDT, allowing reference to a package of definitions for the contained ModuleClasses and device definitions. Can be referenced when extending ModuleClasses. It has two possible uses: to select the scope of a technology domain, or to set the scope of a use case domain (like Home, SmartGrid, etc.).
Device	Physical, addressable, identifiable appliance/sensor/actuator.
Sub-Device	A device (usually one of several) which may be embedded in a Device and/or is addressed via another Device.
ModuleClass	Specification of a **single** service with one or more service methods, the involved abstracted data model and related events. The expectation is that each separate service which may be used in many kinds of Devices (like PowerON/OFF, Open/Close, etc.) will be described by a ModuleClass which can be re-used in many Device definitions.
Module	Instantiation of a ModuleClass for a specific Device or SubDevice.

5.1.3.3 Overview

Various details about recommended structure for SDTs are described in the next clauses. The key point to keep in mind is that HGI sought a compromise between, at the one extreme, complete flexibility (which could describe any device, of any complexity) and, at the other extreme, a rigid structure which could be 100 % validated and lead to validated software APIs.

A major decision, facilitating validation of code and signalling, was to describe services (functionality) of devices in terms of ModuleClasses made up of combinations of three kinds of elements:

a) DataPoints which can be read/written;

b) Actions which consist of more complex sequences of operations;

c) Events which can be signalled ("published") by devices asynchronously.

This structure shown in figure 5.1.3.3-1 and is given in more detail in figure 5.1.3.4-1.

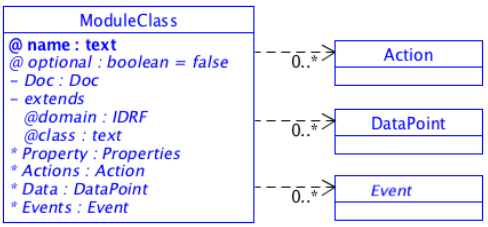

Figure 5.1.3.3-1: Diagram describing device functionality in terms of Actions, DataPoints, Events

5.1.3.4 Structure

The following UML diagram presents an overview of the structure (elements) of every SDT which is conformant with these guidelines. As implied in the above descriptions, there can be many different choices of the details of a SDT, each one optimized for a particular market segment and the types of devices used in that market segment. Obviously an unnecessary proliferation is counter-productive, but as long as each SDT conforms to the structure shown below then it will be possible with little or modest effort for software applications to be adapted accordingly.

The UML diagram below is in a sense **the** meta-format **for** different possible Smart Device Templates (XSDs) **for** device descriptions (XMLs) **of** real devices - sorry about that.

The syntax used in the diagram to model an XML Schema Definition (XSD) as an UML diagram follows the following approaches: Design XML schemas using UML and UML For W3C XML Schema Design.

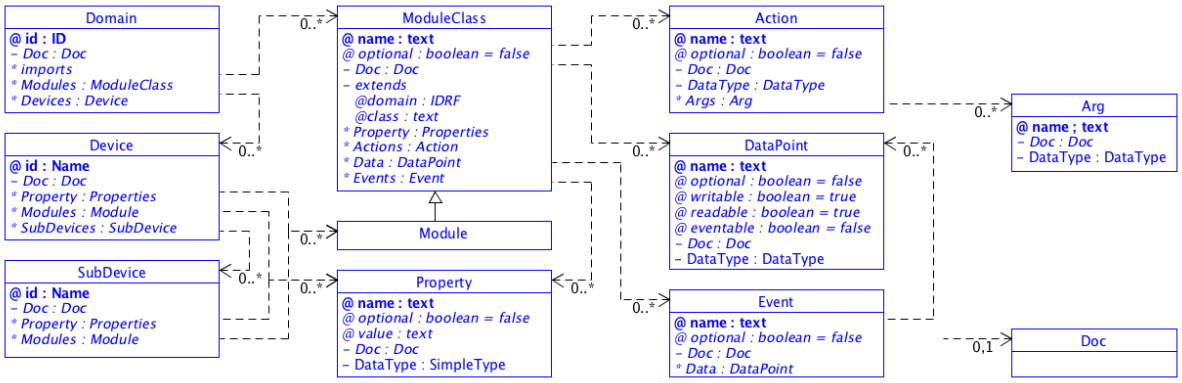

Figure 5.1.3.4-1: UML Diagram providing an overview of the SDT

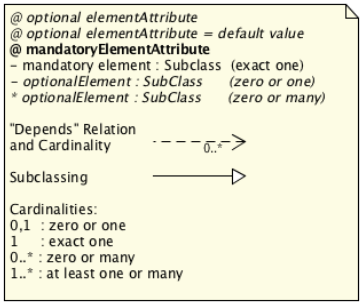

Figure 5.1.3.4-2: Annotations for UML Diagrams in SDT

Several constraints or design decisions were involved in creating the above template, which are describe below at the appropriate Element.

5.1.3.5 "Domain" element

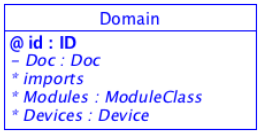

Figure 5.1.3.5-1

The "Domain" element allows labeling of different SDT templates for different technologies and/or industry segments ("verticals"): for example eHealth and Building Management might prefer quite different detailed structures/templates. This also helps keep information in human-friendly and manageable blocks. It is assumed that there will be multiple "SDT Templates" and some of them may be completely proprietary.

It can also be used to collect all specified ModuleClasses and Devices in one referencable logical group.

5.1.3.6 "Device" and "SubDevice" element

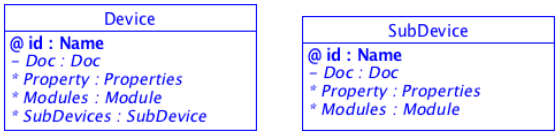

Figure 5.1.3.6-1

The "Device" was initially thought of as the representation of "the basic things we are trying to model" and can still be considered so. However, after discussion with various SDOs, it was decided to add also "sub-devices". That is, there is one level of hierarchy to allow modeling of e.g. a set of independent energy monitoring plugs in a single addressable power-extension-block. (Other SDOs might consider it more appropriate to use a recursive sub-sub-sub ... device definition). Note that all the different devices which one needs to model within a Domain are composed of one or more Modules.

5.1.3.7 "Module" element(s)

"Module" elements are basically constraints or templates for how to model functionality of real things/appliances/devices within the Domain. There could be an infinite number of possible functionalities, however it is recommended to identify a not-too-large selection of them as generic examples (called "ModuleClasses", see clause 5.1.3.8) and allow for additional proprietary extensions. In a particular Domain there will be one Module for each of the agreed ModuleClasses plus additional ones for each extension of a ModuleClass.

The advantage of identifying a subset of generic "ModuleClasses" is that any suitable high-level software would then be able to "parse" the generic functionality for all compliant appliances, even if the proprietary parts could not be interpreted by the software.

Every "Device" can then be described by a collection of "Modules" (functionality). In the simplest examples, where there are no extensions needed, each ModuleClass has exactly one "child" Module ... in such cases the software developer can consider the two terms to be the same.

The relationship between a ModuleClass and a Module is very similar to the specification of a class and an instantiated object in an object oriented programming language.

5.1.3.8 "ModuleClass" element(s)

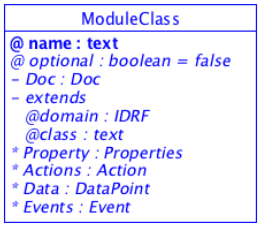

Figure 5.1.3.8-1

The set of "ModuleClasses" is defined at the "Domain" level. Each one describes some functionality (services). In principle there could be an infinite number of ModuleClasses (as noted above), for every kind of functionality found in UPnP, ZigBee and all the other automation protocols ... However that would **not** simplify the job of software developers at all! Therefore, HGI recommends that a finite and convenient number of prototypical ModuleClasses are re-used as much as possible (within a Domain at least).

Typical ModuleClasses might be equivalent to "power ON/OFF", "Open/Close", "PanUP/DOWN", "ReadTemperature", etc. Those examples make it apparent that various read/write usage of parameters, invoking of actions and waiting for events might be needed in the different ModuleClasses, and a guideline for those structures is explained in clauses 5.1.3.9 to 5.1.3.13.

5.1.3.9 "Property" Elements

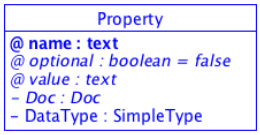

Figure 5.1.3.9-1

"Property" elements are used to append to Devices and ModuleClass elements with arbitrary additional information. For Devices it would be very common for a manufacturer to want to add into the XML file which is describing the device such information as Manufacturing Site, Date of Manufacture, Certification Code, Energy Label Code, compatible LAN technology, URL for the device handbook, physical limits of operation environments, etc. Some of that information might in some devices be available by reading a specific device DataPoint, however even if it cannot be read from the device then at least it can be noted in the device's XML description. Examples for organizations that specify these kind of added "Property" information are eCl@ss and UNSPSC (United Nations Standard Products and Services Code).

Since the Properties are highly varied, depending on industry segment, no attempt is made in the SDT to constrain the options: however it is highly recommended to provide software-developer-friendly information in the DOC field of each Property.

5.1.3.10 "ModuleClass" - "DataPoint" element

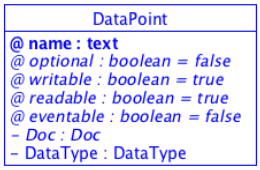

Figure 5.1.3.10-1

A "DataPoint" element represents an aspect of a Device which can be read/written to, and forms part of a device's data model. Manipulating DataPoints is the most common way of controlling Devices. Each DataPoint has an associated "type" (e.g. simple integer/real numbers, string of text, struct, or arrays thereof) which facillitates data integrity. Note that all RESTful systems (e.g. CoAP) only use DataPoint operations, so the mapping of a data models using an SDT into RESTful applications is easy.

However, DataPoints are not the *only* way of controlling devices or reading information from devices, so further "Actions" and "Events are described next.

5.1.3.11 "ModuleClass" - "Actions" element

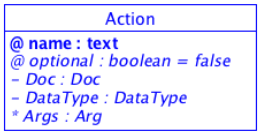

Figure 5.1.3.11-1

"Action" elements are an efficient way of describing arbitrary sequences of operations/methods; these are very common in automation. Typical example include "FactoryReset", and "AutoCalibrate". Actions preserve transaction integrity by putting together all the parameters ("args", see clause 5.1.3.12) with the method which checks and executes them, in one step.

Note that systems which rely on RESTful operations need to carry out such complex *setup-parameters-then-do-action* by first using (several) DataPoint operations to "load" the parameters to the device and then do a DataPoint operation to manipulate the "start operation NOW" action.

5.1.3.12 "Module Class" - "Action" - "Args"

Figure 5.1.3.12-1

The "Args" element represents the parameter information which a device needs to carry out a required "Action".

5.1.3.13 "ModuleClass" - "Event" element

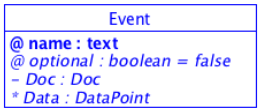

Figure 5.1.3.13-1

"Event" elements are needed for automation protocols which "push" information, instead of relying on polling by the software application. A typical example would be a "SensorAlert" where a window sensor immediately transmits a change of its state from "closed" to "open", which could be used in a burglar alarm application, needs to be ready to accept such information immediately, and not wait for a regular polling of the device.

5.1.3.14 "Doc" element

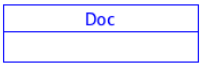

Figure 5.1.3.14-1

"Doc" elements (for all the above Elements) are very important to help understand the software-readable information for specific devices and services. They contain the human-readable information. Many automation protocols describe every possible operation in a comprehensive specification, however SDT is designed to include the relevant information at the "point of use" for the software developer, inside the SDT (and XML files based on it).

5.1.3.15 "DataType" element

The DataType can be simple integers or string text, or rather complex, as shown in figure 5.1.3.15-1.

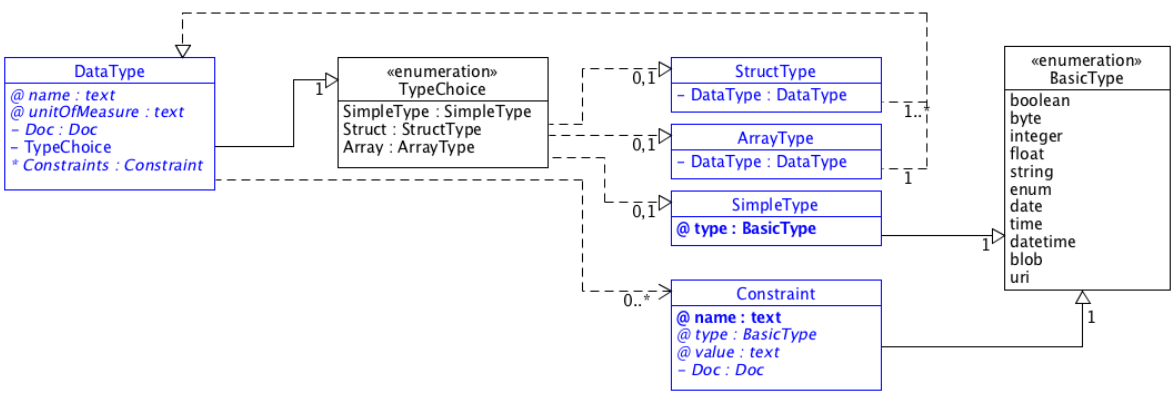

Figure 5.1.3.15-1

5.1.3.16 "DataType" - "unitOfMeasure"

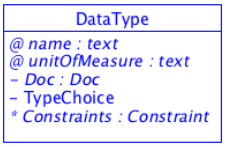

Figure 5.1.3.16-1

Before considering the type of data in detail, there is the option to label the data with the units of measurement. A "Temperature" measurement is meaningless until the units Kelvin, Celcius, Fahrenheit etc are known. Because of the extreme variety of units, a string field is the default annotation method, although of course a SDO could decide to reference a standardized list of units.

Two organizations that provide good definitions for the unit of measure are the "UCUM organization" and "QUDT - Quantities, Units, Dimensions and Data Types Ontologies".

5.1.3.17 "DataType" - "TypeChoice" Element

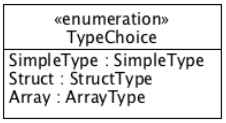

Figure 5.1.3.17-1

The "TypeChoice" element is required for syntactic reasons in the UML diagram and the choice from the enumerated list simply designates the complexity of the following DataType.

5.1.3.18 "DataType" - "SimpleType" Element

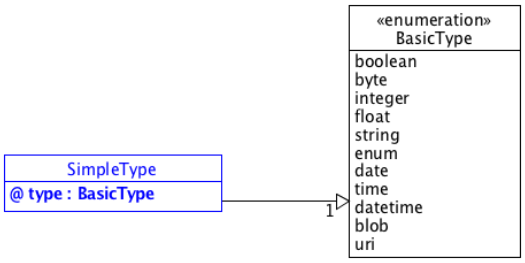

Figure 5.1.3.18-1

The "SimpleType" element is required in order for software to understand the format of the associated data, e.g. are the bytes an integer or real value? HGI decided based on practical experience to include some specific types which are slightly more complex:

1) the (technically redundant) options of "date" and "time"- to avoid problems which can arise interpreting a datetime value;

2) "url" because it is expected to become extremely common to provide links to other data sources;

3) the "blob" type to represent binary data of arbitrary structure.

5.1.3.19 "DataType" - "Constraint" Element

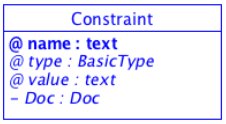

Figure 5.1.3.19-1

The "Constraint" element is an optional element allowing the manufacturer to provide constraints on the permitted values of measured data or input parameters. It can significantly improve the reliability of software and validation of transmitted data.

5.1.3.20 "DataType" - "Array" Element

Figure 5.1.3.20-1

The "Array" element is provided for defining lists of data; the definition is recursive so that multi-dimensional arrays can be described. Note that a Constraint can be used to provide limits on Array size.

5.1.3.21 "DataType" - "StructType" Element

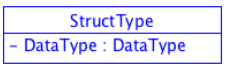

Figure 5.1.3.21-1

The "StructType" element can be used to represent an ordered list of diverse DataTypes, which are represented by the Name attribute of each DataType, and can be used recursively.

5.1.3.22 A very simple SDT example

ln the ideal case, a large organization or SDO would define a widely-applicable set of ModuleClasses, each of which could be used as needed to compose the description of a complex device. In order to show the appoach, this clause will create a few example ModuleClasses based on - or inspired by - features in the Echonet Lite protocol. Please note that the examples shown in the present document are very "cut down" and by no means represent a true representation of Echonet Lite.

The Echonet Consortium has standardized their specifications within IEC/ISO (IEC 62394 [i.11], ISO/IEC 24767-1 [i.12], ISO/IEC 24767-2 [i.13], IEC 62480 [i.14], ISO/IEC 14543-4-1 [i.15], ISO/IEC 14543-4-2 [i.16], IEC 62457 [i.17]) and they provide a comprehensive collection of various types of home appliances relevant to SmartGrid applications as ECHONET Device objects (see https://echonet.jp/spec_object_rf_en/).

For the example in the present document, to show re-use of ModuleClass definitions, two complex devices are "invented" which have some common features and hence could be expected to both use some of the same ModuleClasses: an air conditioner and a washing machine.

Table 5.1.3.22-1

Functionality	Air Conditioner	Washing Machine
operationStatus	operates on/off	operates on/off
measuredCumulativePowerConsumption	the cumulative power consumption	the cumulative power consumption
installationLocation	this sets/reads a string text describing the location (room) of the air-conditioner.	this sets/reads a string text describing the location (room) of the washing machine.
onTimerSetting	This sets/reads the on/off timer	This sets/reads the on/off timer
statusDoor	(This function is not applicable)	This reads whether the door of the washing machine is open or closed.

Based on the simplified example above, the two appliances will need the ModuleClasses below:

- *air-conditioner*: operationStatus, measuredCumulativePowerConsumption, installationLocation, setTimer, statusDoor;
- *washing-machine*: operationStatus, measuredCumulativePowerConsumption, installationLocation, setTimer.

```
<ModuleClass name="operationStatus">
    <Data>
        <DataPoint name="operationStatus" writable="true">
            <Doc>This property sets/reads the ON/OFF status.</Doc>
                <DataType>
                    <SimpleType type="boolean"/>
                </DataType>
        </DataPoint>
    </Data>
    <Events>
        <Event name="operationStatus">
        </Event>
    </Events>
</ModuleClass>

<ModuleClass name="measuredCumulativePowerConsumption">
    <Data>
        <DataPoint name="measuredCumulativePowerConsumption" writable="false">
            <Doc>This indicates cumulative power consumption of the device in increments of
0.001kWh.</Doc>
            <DataType>
                <SimpleType type="integer"/>
            </DataType>
        </DataPoint>
    </Data>
</ModuleClass>

<ModuleClass name="installationLocation">
    <Data>
        <DataPoint name="installationLocation" writable="true">
            <Doc>This property indicates the installation location</Doc>
            <DataType>
                <SimpleType type="string"/>
            </DataType>
        </DataPoint>
    </Data>
    <Events>
        <Event name="installationLocation"> </Event>
    </Events>
</ModuleClass>

<ModuleClass name="onTimerSetting">
    <DataPoint name="onTimer" writable="true">
        <Doc>Timer value (HH:MM)</Doc>
        <DataType>
            <SimpleType type="time"/>
        </DataType>
    </DataPoint>
</ModuleClass>
<ModuleClass name="statusDoor">
    <DataPoint name="stausDoor" writable="false">
        <Doc>This reads the open=true or closed=false status of a door </Doc>
        <DataType>
```

```
            <SimpleType type="boolean"/>
        </DataType>
    </DataPoint>
</ModuleClass>
```

The Example1.SDT now looks like this:

Example1.SDT	
	Namespace information
	Modules (contains ModuleClasses) • operationStatus • measuredCumulativePowerConsumption • installationLocation • onTimerSetting • statusDoor

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Example1 SDT inspired by some Echonet Lite examples -->
<Domain xmlns="http://homegatewayinitiative.org/xml/dal/3.0"
    xmlns:xi="http://www.w3.org/2001/XInclude"
    id="example1.SDT">
<Modules>
<!-- Various examples for module classes -->
<ModuleClass name="operationStatus">
    <Data>
        <DataPoint name="operationStatus" writable="true">
            <Doc>This property sets the ON/OFF status.</Doc>
                <DataType>
                    <SimpleType type="boolean"/>
                </DataType>
        </DataPoint>
    </Data>
    <Events>
        <Event name="operationStatus">
        </Event>
    </Events>
</ModuleClass>

<ModuleClass name="installationLocation">
    <Data>
        <DataPoint name="installationLocation" writable="true">
            <Doc>This property indicates the installation location</Doc>
            <DataType>
                <SimpleType type="string"/>
            </DataType>
        </DataPoint>
    </Data>
    <Events>
        <Event name="installationLocation"> </Event>
    </Events>
</ModuleClass>
<ModuleClass name="measuredCumulativePowerConsumption">
    <Data>
        <DataPoint name="measuredCumulativePowerConsumption" writable="false">
            <Doc>This indicates cumulative power consumption of the device in increments of
0.001kWh.</Doc>
            <DataType>
                    <SimpleType type="integer"/>
            </DataType>
        </DataPoint>
    </Data>
</ModuleClass>

<ModuleClass name="onTimerSetting">
    <DataPoint name="onTimer" writable="true">
        <Doc>Timer value (HH:MM)</Doc>
        <DataType>
            <SimpleType type="time"/>
        </DataType>
    </DataPoint>
</ModuleClass>
    </Modules>
</Domain>
```

A manufacturer could now describe the behaviour of a specific device called "Air Conditioner" by the following XML file referencing the Example1.SDT:

- // INCLUDE XML FILE here, for a washing machine ///

The Example1.SDT could in future, in this hypothetical explanation, also be expanded by adding the air-conditioner as a template device, allowing other manufacturers to use and extend the description for their special models:

Example2.SDT	
	Namespace information
	Modules (contains ModuleClasses) • operationStatus • installationLocation • onTimerSetting • measuredCumulativePowerConsumption • statusDoor
	Generic Devices (can be used as protogenitors) • Air-Conditioner

5.1.4 ECHONET/ECHONET Lite (ECHONET Consortium)

The ECHONET is an open communication proximity network protocol that supports smart house and Home Energy Management System. The specification of the ECHONET is determined by the ECHONET Consortium [i.8]. ECHONET Lite has been released as a modification of the ECHONET mainly by removing the underlying network layer technology on the ECHONET [i.9].

ECHONET Consortium specify the information model called "Device Object" [i.10]. Device Object represents the property sets for devices. Over 90 types of devices for device object have been currently defined and been updated from time to time. An example of ECHONET device object is shown as figure 5.1.4-1.

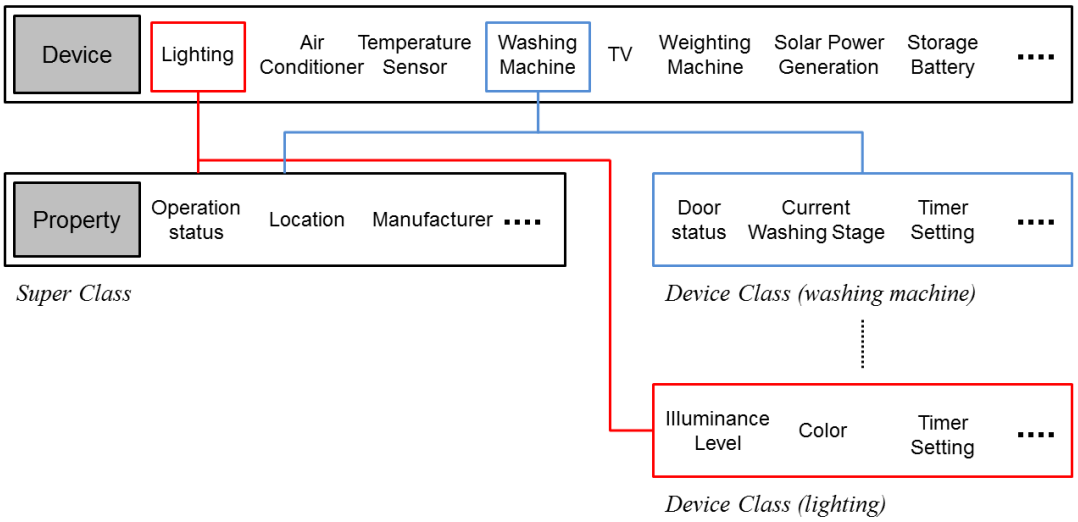

Figure 5.1.4-1: Information Model in ECHONET Device Object

As shown in figure 5.1.4-1, the Device Object consist of "Super Class" which represents common properties among all devices and "Device Class" which represents specific properties for its own device type (e.g. washing machine, lighting, etc.). Each property has a flag which indicates mandatory or optional as the ECHONET certification.

5.1.5 OIC (Open Interconnect Consortium)

OIC system is an interoperable framework for interaction between devices within local or remote networks. In OIC architecture, entities in physical world for e.g. bulb are represented as resources with their unique URIs and supported interfaces that enable RESTful operations on these resources. OIC Smart Home profile defines a set of device types and the set of minimum resource types that a device type has to support. Each resource type defines a set of minimum properties that resource type should support. All resources should support a minimum set of properties called common properties. These common properties are used for common functions like discovery. Figure 5.1.5-1 shows the OIC resource model especially for refrigerator device type. The arrows show the minimum resources that the refrigerator device type should support and the minimum properties that each of the resources should support.

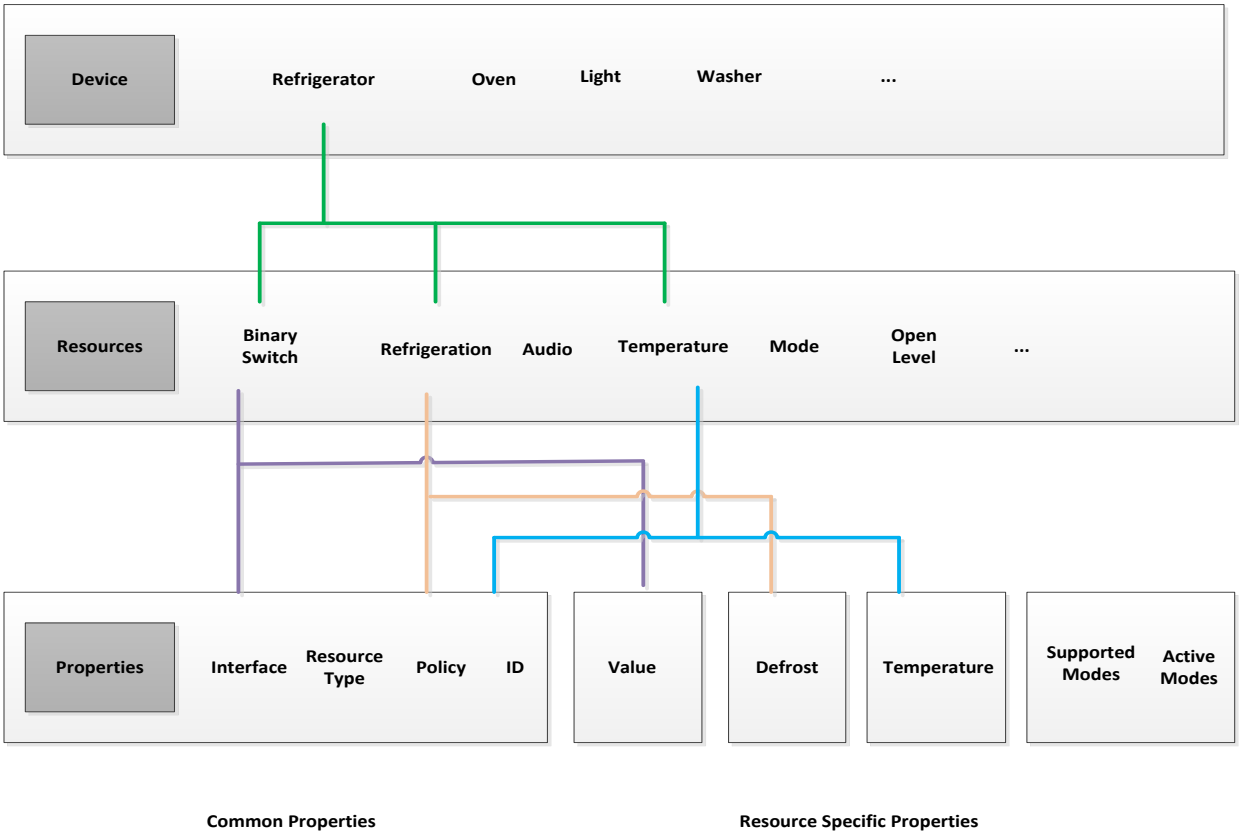

Figure 5.1.5-1: OIC Resource Model

As can be seen from figure 5.1.5-1, the OIC resource model is a RESTful model which defines minimum set of resources for each smart home device type. As the need for new profiles arise additional resources can be defined and hence the model is scalable. Also each vendor can customize the model for any device type by adding function specific resources and properties. Resources are re-usable across device types. A set of properties common between resources is identified as common properties. Since oneM2M architecture is also a RESTful architecture, the OIC resource model maps very well architecturally to oneM2M. Also all the OIC Smart Home resources are semantically aligned with UPnP smart home resources.

5.2 Design Principle of Abstract Information Model

5.2.1 Introduction

This clause defines a design principle of the abstract information model for the home domain appliance. Based on analysis works in the clause 5.1, design principles of abstract information model may be specified. (That is, serveral design principles can be discussed in the present document to discovery and/or make an optimum abstract information model in the home domain environment.)

5.2.2 Design Principle: Approach 1

Based on the analysis work for the AllJoyn (clause 5.1.1), HomeKit (clause 5.1.2), HGI Smart Device Template (clause 5.1.3) and ECHONET (clause 5.1.3), design principles of the abstract information model are as follows:

- Each device type has the abstract information model of its own.
- Vendor-specific function is needed for un-predefined characteristic(s).
- Common characteristics are needed to avoid duplicating the description.
- Home appliances shall be composed of one or more characteristic(s).

Based on the above principles, the basic abstract information model is designed as shown in figure 5.2.1-1.

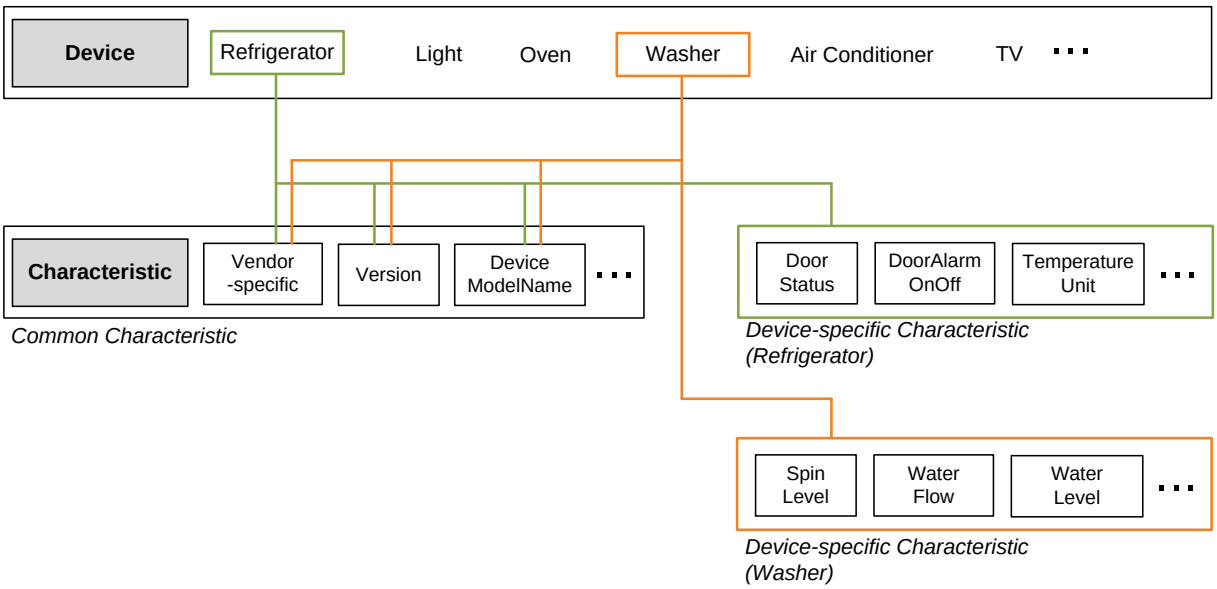

Figure 5.2.2-1: Design Principle of Abstract Information Model

- 'Characteristic' specifies pre-defined device characteristics which mean an atomic fuction (such as version, device model name, door status). Since some characteristic(s) may present in all in all device, the characteristic is clarified as common and device-specific characteristic in order to avoid duplicating the description for every device.
- 'Device' which represents physical home appliance (such as TV, Air Conditioner, Refrigerator) is composed of one or more pre-defined characteristic(s).

5.2.3 Design Principle: Approach 2

The differences from the approach 1 are as follows:

- The approach 2 defines 'Service' that may have one or more characteristic(s).
- The approach 2 does not categorize 'Characteristic', but 'Service' into common, shared and device-specific.

The concept of 'Service' is similar to the AllJoyn concept of 'Interface', the HomeKit concept of 'Service' and the SDT concept of 'ModuleClasses'. This would help to increase modularity for functions and thus enhance resource efficiency.

Design principles of the abstract information model are as follows:

- Each device type has the abstract information model of its own.
- Home appliances shall be composed of one or more service(s).
- Services is categorized into three types (common, shared and device-specific)

- Common services are needed to avoid duplicating the description.
- Shared services are needed to enhance the resource efficiency by re-using it.
- Device-specific services are needed to support specific functions for specific devices.
- A service is composed of one or more pre-defined characteristic(s).
- A characteristic can be used to represent either an attribute or a behavior.

NOTE: When a characteristic represents a behaviour, a service can be operated by just updating the characteristic.

The abstract information model structure based on the design principles above is illustrated in the figure 5.2.2-1.

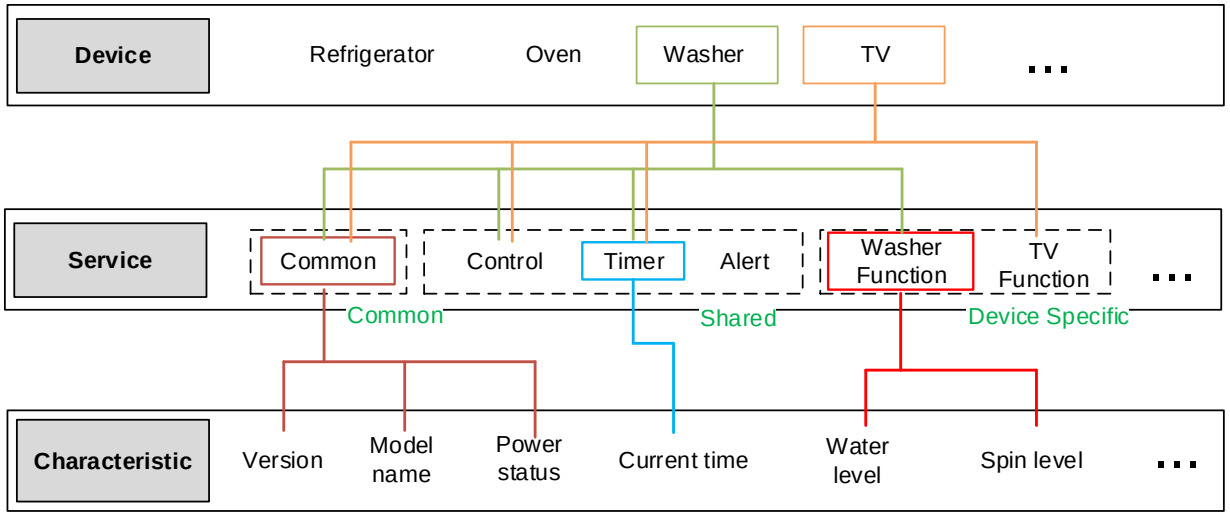

Figure 5.2.3-1: Abstract Information Model

5.2.4 Design Principle: Approach 3

In order to make the oneM2M Abstact Information Model for Home Appliances more interoperable with external technologies, it would be a desirable way to accommodate existing information models as much as possible. In this approach, a new principle applying the basic concept of AllJoyn and SDT is introduced.

The differences from the approach 2 are as follows:

- The approach 3 supports 'Sub-device'.
- The approach 3 newly defines 'Device characteristic' that indicates properties of a device.
- 'Service' is composed of 'Method', 'Service characteristic' and 'Event'.

Design principles of the abstract information model are as follows:

- Each home appliance type has the abstract device model of its own.
- Device:
  - Device is the definition of a physical device that may contain optional embedded SubDevices.
  - For each physical device on the network at least one device should be defined.
  - If the device is a simple device, it does not include further devices. On the other hand, if the device is a compound device, the device defines devices for each of the identifiable embedded devices.
  - Devices may define own Services or refer to pre-defined Services.
  - Device information can be indicated by Device Characteristic.

- Service:
  - Service represents the concept of a reusable module that defines the Method, Service Characteristics and Events for a single functionality of a device.
  - A connected device may contain or refer to single or multiple Services to specify its inherent services it exposes.
- Device Characteristic:
  - Device Characteristic are used to append to Devices with arbitrary additional information. It would be very common information such as date of manufacture, product brand or country of production.
- SubDevice:
  - SubDevices are optional components of a device. They represent SubDevices inside a device.
  - SubDevices may define their own Services or extend pre-defined Services.
- Method:
  - Method are an way of describing arbitrary sequences of operations.
- Service Characteristic:
  - Service Characteristics mean data members. These are accessed by CRUD operation or by buit-in get/set methods.
- Event:
  - Event is needed for providing asynchronous event notification (Push).

The abstract information model structure based on the design principles above is illustrated in the figure 5.2.3-1.

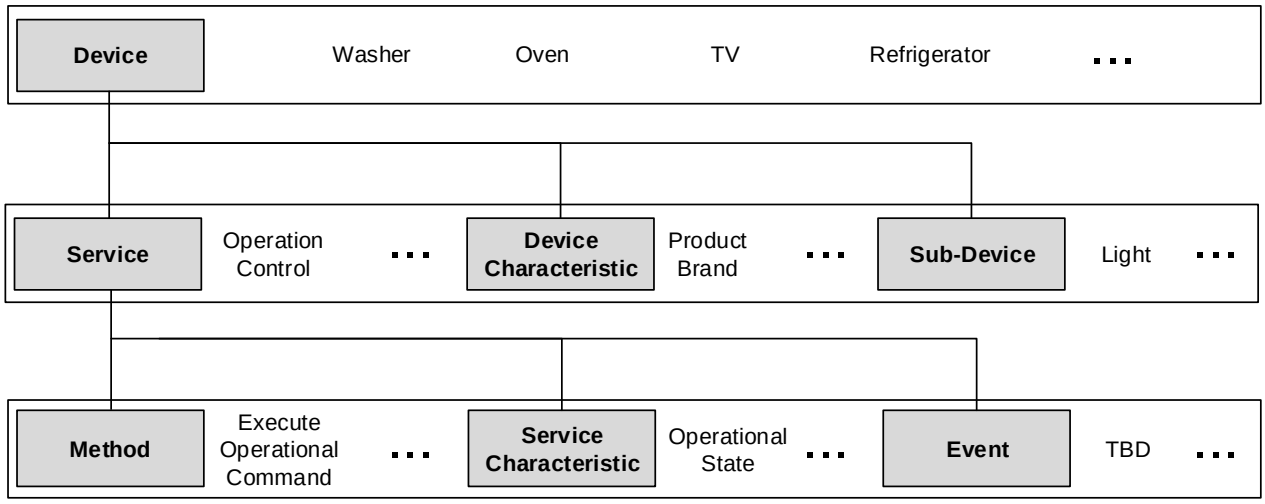

Figure 5.2.4-1: Abstract Information Model

5.2.5 Design Principle: Approach 4

Approach 4 is to use HGI SDT, as it is, that is explained in clause 5.1.3. Since HGI SDT is well desinged to define important facts for abstract information model and already reviewed by technology providers, thus, SDT can be one of approach for defining oneM2M information model.

The principle of information model using SDT is again illustrated for reader's convenience in the figure 5.2.4-1.

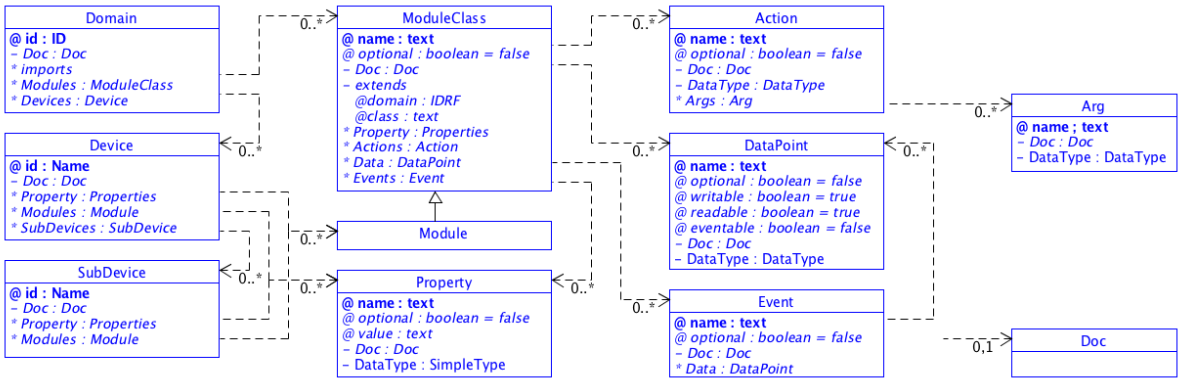

Figure 5.2.5-1: Principle of Information Model using SDT

5.3 Define Abstract and Describe Specific Abstract Information Model

5.3.1 Abstract Information Model: Approach 1

5.3.1.1 Introduction

This approach is based on the 'Design principle: Approach 1' (clause 5.2.1). That is, this clause provides details on the 'Characteristic' and 'Device' for the abstract information model of the home domain. Here, the data type is defined in [i.7].

- Common Chracteristic Sets.

Since some characteristics in all home domain devices may have same values, the common characteristic sets are considered to avoid duplicating the description.

Table 5.3.1.1-1: Common Characteristics

Characteristic Name	Description
deviceFirmwareVersion	Firmware version
deviceFirmwareID	Firmware ID
deviceType	Device type. Each device type is defined as follows: • 101: refrigerator • 201: washer • 301: oven • 401: airconditioner • 501: robot cleaner • 601: smart meter
deviceAliasName	Device alias name
deviceSubModelName	Device sub-model name
nation	Nation code
deviceModelName	Device model name
deviceID	Device ID
powerStatus	Power status of the device. Power status is defined as follows: • 0: off • 1: on
representedPower	Instantaneous power consumption of the device.
usedEnergy	Cumulative energy consumption of the device.

- Device Sets

In the device sets, device-specific characteristic(s) as well common characteristic(s) are defined. Among home domain devices, the washer is considered for representation.

5.3.1.2 Washer

Table 5.3.1.2-1: Abstract Information Model of Washer

Characteristic Type	Characteristic Name	Data Type	Value	Permission	Description
Common	*deviceFirmwareID*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceFirmware Version*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceType*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceAlias Name*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceSub ModelName*	xs:string	-	RO	See table 5.3.1-1
Common	*nation*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceModel Name*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceID*	xs:string	-	RO	See table 5.3.1-1
Common	*powerStatus*	xs:boolean	-	RW	See table 5.3.1-1
Common	*representedPower*	xs:double	-	RO	See table 5.3.1-1
Common	*usedEnergy*	xs:double	-	RO	See table 5.3.1-1
Device-specific	*totalTime*	xs:string	-	RO	Remain time + Reserve time (HHMM)
Device-specific	*remainTime*	xs:string	-	RO	Remain time (HHMM)
Device-specific	*reserveTime*	xs:string	-	RO	Reserve time (HHMM)
Vendor-specific	*vendorSpecific*	xs:anySimpleType	-	RW	The characteristic contains vendor-specific information.

5.3.1.32 Refrigerator

Table 5.3.1.3-1: Abstract Information Model of Refrigerator

Characteristic Type	Characteristic Name	Data Type	Value	Permission	Description
Common	*deviceFirmwareID*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceFirmware Version*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceType*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceAlias Name*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceSub ModelName*	xs:string	-	RO	See table 5.3.1-1
Common	*nation*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceModel Name*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceID*	xs:string	-	RO	See table 5.3.1-1
Common	*powerStatus*	xs:boolean	-	RW	See table 5.3.1-1
Common	*representedPower*	xs:double	-	RO	See table 5.3.1-1
Common	*usedEnergy*	xs:double	-	RO	See table 5.3.1-1
Device-specific	*freezerTemp*	xs:integer	-	RW	Temperature of freezer
Device-specific	*fridgeTemp*	xs:integer	-	RW	Temperature of fridge
Vendor-specific	*vendorSpecific*	xs:anySimpleType	-	RW	The characteristic contains vendor-specific information.

5.3.1.4 Smart Meter

Table 5.3.1.4-1: Abstract Information Model of Smart Meter

Characteristic Type	Chracteristic Name	Data Type	Value	Permission	Description
Common	*deviceFirmwareID*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceFirmware Version*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceType*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceAlias Name*	xs:positiveInteger	-	RO	See table 5.3.1-1
Common	*deviceSub ModelName*	xs:string	-	RO	See table 5.3.1-1
Common	*nation*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceModel Name*	xs:string	-	RO	See table 5.3.1-1
Common	*deviceID*	xs:string	-	RO	See table 5.3.1-1
Common	*powerStatus*	xs:boolean	-	RW	See table 5.3.1-1
Common	*representedPower*	xs:double	-	RO	See table 5.3.1-1
Common	*usedEnergy*	xs:double	-	RO	See table 5.3.1-1
Device-specific	*instantaneous Consumption*	xs:double	-	RO	Measured instantaneous consumption.
Device-specific	*cumulative Consumption*	xs:nonNegativeInteger	-	RO	Measured cumulative consumption.
Device-specific	*limitOf Consumption*	xs:double		RW	Cofigured limitation of consumption.
Device-specific	*instantaneous Generation*	xs:double	-	RO	Measured instantaneous generation.
Device-specific	*cumulative Generation*	xs:nonNegativeInteger	-	RO	Measured instaneous generation.
Device-specific	*limitOf Generation*	xs:double	-	RW	Cofigured limitation of generation.
Device-specific	*objective*	xs: nonNegativeInteger	0: electricity 1: gas 2: water	RO	Objective of meter
Device-specific	*scope*	xs: nonNegativeInteger	0: whole house 1: breaker 2: room 3: device	RO	Scope of meter
Vendor-specific	*vendorSpecific*	xs:anySimpleType	-	RW	The characteristic contains vendor-specific information.

5.3.2 Abstract Information Model: Approach 2

This clause is based on the 'Design principle: Approach 2' (clause 5.2.2) and provides details on the 'Service', 'Characteristic', and 'Device' for the abstract information model of the home domain. The data type is defined in [i.7].

- Common Service:
  - This is the only service that includes some common characteristics and/or functions that all home appliances should have.
  - The objective of defining common service is to avoid duptlicating the description.

Table 5.3.2-1: Common Service

Characteristic Name	Description
deviceType	Device type. Each device type is defined as follows: • 101: refrigerator • 201: washer • 301: oven • 401: airconditioner • 501: robot cleaner • 601: smart meter • 700999: User defined device
deviceID	Device ID
manufacturerBrand	Brand name

- Shared Services:
  - Some home appliance devices may use this services or not.
  - The objective of defining shared services is to avoid duplicating the description and maximize the efficiency in terms of modularity.

Table 5.3.2-2: Shared Service: Timer

Characteristic Name	Description
currentTime	Current time
targetTimetoStart	The time when the device is expected to start its operation
targetTimetoStop	he time when the device is expected to stop its operation
targetDuration	Target duration of the operation

Table 5.3.2-3: Shared Service: Control

Characteristic Name	Description
operationState	Current operational state • 0: off; • 1: idle; • 2: working; • 3: paused

- Device-specific Services:
  - This services is defined, if necessary, in specific device information models.

5.3.2.1 Washer

Table 5.3.2.1-1: Abstract Information Model of Washer

Service Type	Characteristic Name	Data Type	Value	Permission	Description
Common	*deviceType*	xs:positiveInteger	-	RO	See table 5.3.2-1
Common	*deviceID*	xs:string	-	RO	See table 5.3.2-1
Common	*manufacturer brand*	xs:string	-	RO	See table 5.3.2-1
Shared_Timer	*currentTime*	xs:positiveInteger	-	RO	See table 5.3.2-2
Shared_Timer	*targetTimetoStart*	xs:positiveInteger	-	RW	See table 5.3.2-2
Shared_Timer	*targetTimetoStop*	xs:positiveInteger	-	RW	See table 5.3.2-2
Shared_Timer	*targetDuration*	xs:positiveInteger	-	RW	See table 5.3.2-2
Shared_Control	*operationState*	xs:positiveInteger	-	RW	See table 5.3.2-3
Device-specific Washer function	*totalTime*	xs:string	-	RO	Remain time + Reserve time (HHMM)
Device-specific Washer function	*remainTime*	xs:string	-	RO	Remain time (HHMM)
Device-specific Washer function	*reserveTime*	xs:string	-	RO	Reserve time (HHMM)
Device-specific Washer function	*waterLevel*	xs:positiveInteger	-	RW	Level of water for a washer device
Device-specific Washer function	*spinLevel*	xs:positiveInteger	-	RW	Level of spin for a washer device

5.3.3 Abstract Information Model: Approach 3

5.3.3.1 Introduction

NOTE: More device and service information models in detail will be defined in oneM2M-TS-0023 [i.18] .

This clause shows the draft version of information models and some service models based on the 'Design principle: Approach 3' as examples.

5.3.3.2 Device Information Model

A. Television

Table 5.3.3.2-1: Device Information Model of Television

Type	Name	Description	Type	Mandatory/Optional
Service	Control	Operational control (e.g. Turn on/off)	-	M
	AudioVolume	Audio volume control (e.g. up/down)	-	M
	TelevisionChannel	Channel control	-	M
	AudioVideoInput	Audio source control	-	O
	VirtualKey	Receiving key input from controller	-	O
	Mouse	Supporting mouse movement	-	O
	Display	Monitor display control	-	M
Device Characteristic	CountryOfProduction	Country that made the device (e.g. Korea)	string	M
	Location	Location of TV (e.g. living room)	string	O
	CorporateBrand	Company that make the device (e.g. LGE)	string	M
	ProductBrand	Name of the device (e.g. LG OLED TV)	string	M
	RemoteControl	Availability of being controlled by remote device (outside of home domain)	boolean	M
Sub-Device	TBD	Sub-device (e.g. LED)	N/A	O

5.3.3.3 Service Information Model

A. Control

Table 5.3.3.3-1: Service Information Model of Control

Type	Name	Description	Type
Method	ExecuteOperationalCommand	Execute an operational command	-
Service Characteristic	OperationalState	Current operational state of an appliance: • * 0: Off, • 1: Idle, • 2: Working, • 3: ReadyToStart, • 4: DelayedStart, • 5: Paused, • 6: EndOfCycle	TBD
	SupportedOperationalStates	Subset of OperationalState (some devices do not support some states)	TBD
	SupportedOperationalCommands	Supported operational commands by the appliance: • * 0: Off, • 1: On, • 2: Start, • 3: Stop, • 4: Pause, • 5: Resume	TBD
Event	TBD		-

B. AudioVolume

Table 5.3.3.3-2: Service Information Model of AudioVolume

Type	Name	Description	Type
Method	UpVolume	Up volume to the next available value.	-
	DownVolume	Down volume to the next available value	-
	ChangeVolume	Change the volume	-
	EnableMute	Enable/Disable volume mute of device.	-
Service Characteristic	Volume	Speaker volume index of the device. Minimum volume is always 0.	TBD
	StepValue	Step value of volume control.	TBD
	MaxValue	Maximum value allowed for Volume.	TBD
	Muted	The state of volume muted.	TBD
Event	TBD		-

C. TelevisionChannel

Table 5.3.3.3-3: Service Information Model of TelevisionChannel

Type	Name	Description	Type
Method	GetChannelList	Retrieve the list of channels supported by a device which has channel functionality	-
	UpChannel	Move up to the next available channel	-
	DownChannel	Move down to the next available channel	-
	ChangeChannel	Change channel using channelId	-
Service Characteristic	ChannelId	Current channel ID	TBD
	TotalNumberOfChannels	Total number of scanned channels.	TBD
Event	TBD		-

D. AudioVideoInput

Table 5.3.3.3-4: Service Information Model of AudioVideoInput

Type	Name	Description	Type
Method	ChangeInputSource	Change the input source with InputSourceId	-
Service Characteristic	InputSourceId	Activated input source id in the supported input source list: • * 0: RGB, • 1: DVI, • 2: HDMI, • 3: Networt	TBD
	SupportedInputSources	Supported input sources for the given device	TBD
Event	TBD		-

5.3.4 Abstract Information Model: Approach 4

Editor's note: The tables in the following clauses are introduced as examples. More ModuleClasses and Device models in detail will be defined in oneM2M TS-0023 [i.18].

5.3.4.1 ModuleClasses

A. operationControl

Table 5.3.4.1-1: operationControl ModuleClass

Doc	OperationControl provides capabilities to get information about operational state of an appliance and to control an appliance with a specific set of commands linked to appliance operational state.			
Action	**Name**	**Doc**	**Argument**	**Return type**
	ExecuteOperationalCommand	Execute an operational command	Name: "command" Type: integer	None
DataPoint	**Name**	**Doc**	**Type**	**Read/Write**
	OperationalState	Current operational state of an appliance: • * 0 --- Off • * 1 --- Idle • * 2 --- Working • * 3 --- ReadyToStart • * 4 --- DelayedStart • * 5 --- Paused • * 6 --- EndOfCycle	Integer	Read only
	SupportedOperationalStates	Subset of OperationalState	Integer Array	Read only
	SupportedOperationalCommands	Supported operational commands by the appliance: • * 0 --- Off • * 1 --- On • * 2 --- Start • * 3 --- Stop • * 4 --- Pause • * 5 --- Resume	Integer Array	Read only
Property	**Name**	**Doc**	**Type**	
	TBD…			
Event	**Name**	**Doc**	**DataPoint**	
	TBD…			

B. AudioVolume

Table 5.3.4.1-2: Service Information Model of AudioVolume

Doc	audioVolume provides capabilities to control and monitor audio volume.			
Action	**Name**	**Doc**	**Argument**	**Return type**
	UpVolume	Up volume to the next available value	None	None
	DownVolume	Down volume to the next available value	None	None
	ChangeVolume	Change the volume	Name: "volume" Type: integer	None
	EnableMute	Enable/Disable volume mute of device.	Name: "mute" Type: boolean	None
DataPoint	**Name**	**Doc**	**Type**	**Read/Write**
	Volume	Speaker volume index of the device. Minimum volume is always 0.	Integer	Read only
	StepValue	Step value of volume control.	Integer	Read only
	MaxValue	Maximum value allowed for Volume.	Integer	Read only
	Muted	The state of volume muted.	Integer	Read only
Property	**Name**	**Doc**	**Type**	
	TBD…			
Event	**Name**	**Doc**	**DataPoint**	
	TBD…			

5.3.4.2 Device Information Model

A. Television

Table 5.3.4.2-1: Device Information Model of Television

Type	Name	Description	Type	Mandatory/ Optional
Module	Control	Operational control (e.g. Turn on/off)	-	M
	AudioVolume	Audio volume control (e.g. up/down)	-	O
	TelevisionChannel	Channel control	-	O
	AudioVideoInput	Audio source control	-	O
	VirtualKey	Receiving key input from controller	-	O
	Mouse	Supporting mouse movement	-	O
	Display	Monitor display control	-	O
Property	CountryOfProduction	Country that made the device (e.g. Korea)	string	O
	Location	Location of TV (e.g. living room)	string	O
	CorporateBrand	Company that make the device (e.g. LGE)	string	O
	ProductBrand	Name of the device (e.g. LG OLED TV)	string	O
	RemoteControl	Availability of being controlled by remote device (outside of home domain)	boolean	O
Sub-Device	TBD	Sub-device (e.g. LED)	-	O

6 Representation in the oneM2M System

6.1 Introduction

This clause describes potentially how the developed abstract information model for a device could be represented in the CSEs of the oneM2M system.

The developed abstract information model for home appliances (see clause 5.3) should be represented somehow as a resource within the oneM2M system. There could be several ways to do so, and they might have different features.

The following clauses describe potential ways of representation and provide the pros and cons of each solution for intercomparison which could then lead to the clause 7.

6.2 Approach 1: Representation Using Dedicated Resource Types

6.2.1 New Resource Type *appliance*

6.2.1.1 Introduction

This clause proposes a new resource type that can represent the oneM2M Information Model defined in the clause 5.3.2.

The <*appliance*> resource represents an application object that is exposed by an M2M Application to enable other M2M Applications to manipulate its object data (e.g. home appliance control data). Manipulation of the resource attributes are notified to the M2M Application then it can perform its application logics and return results. Those attributes to manipulate the application object are pre-defined so that the Originator application and the target application already share the control information and its data formats.

The <*appliance*> resource is a child resource of the <*AE*> resource. The target M2M application (e.g. which hosts home appliance control application) creates the resource under its <*AE*> resource and other M2M Applications discover and manipulates with the proper privileges.

Each instance of <*appliance*> resource is the instance of specializied <*appliance*> resource.

There are two approaches to define <*appliance*> resource type. They are one-level representation and two-level representation.

6.2.1.1 one-level <*appliance*>

All services including common service, shared services and device-specific services defined in clause 5.3.2 are represented as attributes of <*appliance*>. Common service that indicating basic properties of a device are directly mapped to attributes under <*appliance*>. And, shared services and device-specific services are mapped to [sharedChar] and [deviceSpecificChar] respectively because they are variable.

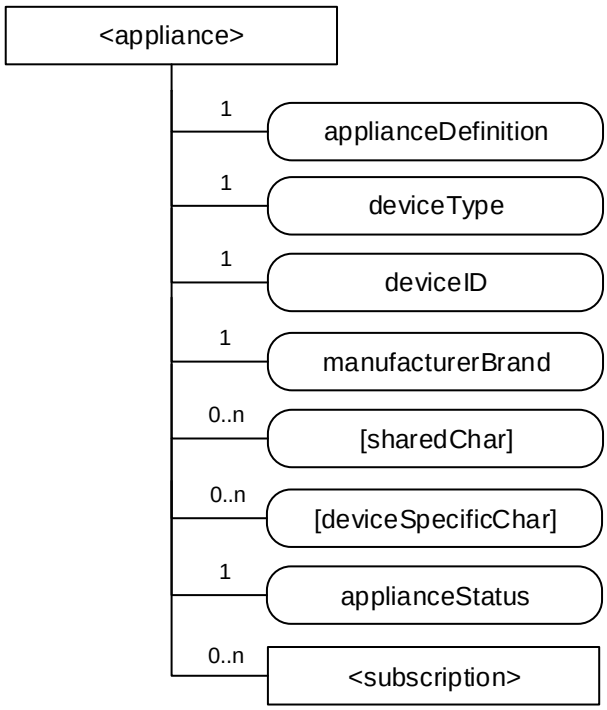

Figure 6.2.1.1-1: Structure of <*appliance*> resource

The <*appliance*> resource should contain the child resources specified in table 6.2.1.1-1.

Table 6.2.1.1-1: Child resources of <*appliance*> resource

Child Resources of <*appliance*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The <*appliance*> resource should contain the attributes specified in table 6.2.1-2.

Table 6.2.1.1-2: Attribute of <*appliance*> resource

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description	<*applianceAnnc*> Attributes
resourceType	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
resourceID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
resourceName	1	WO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
parentID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
expirationTime	1	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
labels	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
creationTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
lastModifiedTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announceTo	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announcedAttribute	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
applianceDefinition	1	WO	Definition of the application that is exposed by this resource (e.g. Washer)	MA
deviceType	1	RO	See clause 5.3.2	OA
deviceID	1	RO	See clause 5.3.2	OA
manufacturerBrand	1	RO	See clause 5.3.2	OA
[sharedChar]	0..n	RW	List of shared attributes which are defined per each specialized <*appliance*> resource. Specialized attributes can have characteristics or behaviors that are defined in oneM2M information model	OA
[deviceSpecificChar]	0..n	RW	List of device-specific attributes with are defined per each specialized <appliance> resource Specialized attributes can have characteristics or behaviors that are defined in oneM2M information model	OA
applianceStatus	1	RO	Status information of the application which is represented as the parent <*AE*> resource (e.g. online, offline)	OA
NOTE 1: Some characteristics such as deviceFirmwareVersion, deviceFirmwareID and deviceID which will be needed in the future are already defined in oneM2M system. So they can be represented in a different way using some kinds of linking attributes				

.

Example A. Resource *washer* (Specialization of one-level <*appliance*>)

The [*washer*] resource is used to share information regarding wahser. The [*washer*] is a specialization of the one level <*appliance*> resource.

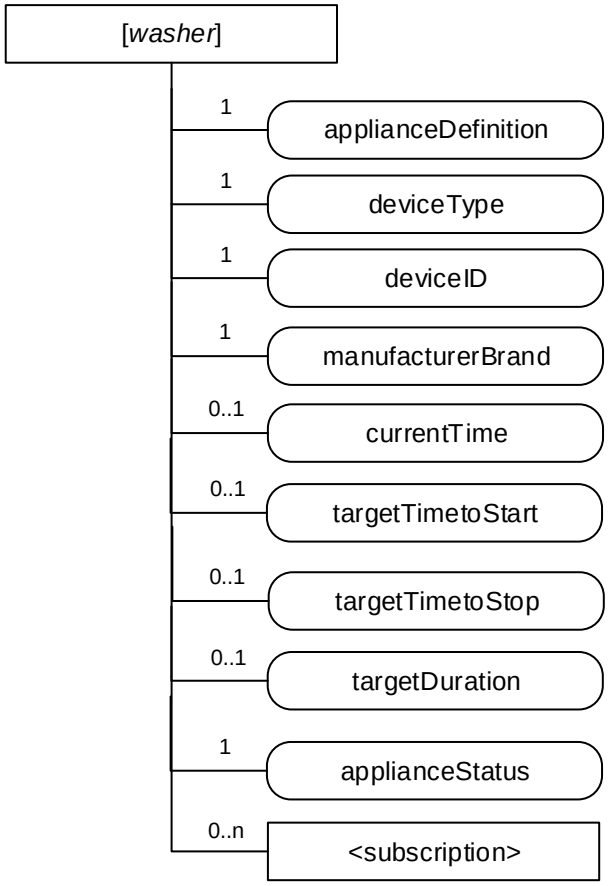

Figure 6.2.1.1-2: Structure of [*washer*] resource

The [*washer*] resource should contain the child resource specified in table 6.2.1.3-3.

Table 6.2.1.3-3: Child resources of [*washer*] resource

Child Resources of <*washer*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The <*washer*> resource should contain the attributes specified in table 6.2.1.3-4.

Table 6.2.1.3-4: Attribute of <*washer*> resource

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description
resourceType	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
resourceID	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
resourceName	1	WO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
parentID	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
expirationTime	1	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
labels	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
creationTime	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
lastModifiedTime	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
announceTo	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description
announcedAttribute	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
applianceDefinition	1	WO	Definition of the application that is exposed by this resource (e.g. Washer)
deviceType	1	RO	Common, see table 5.3.2-1
deviceID	1	RO	Common, see table 5.3.2-1
manufacturerBrand	1	RO	Common, see table 5.3.2-1
currentTime	0..1	RW	Current time
targetTimetoStart	0..1	RW	The time when the device is expected to start its operation
targetTimetoStop	0..1	RW	The time when the device is expected to stop its operation
targetDuration	0..1	RW	Target duration of the operation
applianceStatus	1	RO	Status information of the application which is represented as the parent <*AE*> resource (e.g. online, offline)

As a procedure example, the figure 6.2.1.1-3 illustrates how the [*washer*] resource works for controlling the device.

There are three players in the example. **Remote Controller** (AE) can control or monitor Washer device using [*washer*] resource which is located at the **Gateway** (Hosting CSE) and requested by **Washer** (AE).

- Step-001: Washer (AE) requests AE registration to the Gateway (Hosting CSE).
- Step-002: After checking authority using a Certificate or Service Subscription Profile, the Hosting CSE should create <*AE*> resource under <*CSEBase*> resource.
- Step-003: The Hosting CSE should respond with reasonable parameters in Response message for <*AE*> Create operation.
- Step-004: Washer (AE) requests for creating [*washer*] resource to the Gateway (Hosting CSE).
- Step-005: If the request is from the creator of <*AE*>, the Hosting CSE should create [*washer*] resource under <*AE*>. Otherwise, it rejects.
- Step-006: The Hosting CSE should respond with reasonable parameters in Response message for [*washer*] Create operation.
- Step-007: Washer (AE) requests for creating <subscription> resource preparing to get a notification which will be regarded as a command to actuate the physical device.
- Step-008: After checking privileges, the Hosting CSE should create <*subscription*> resource under [*washer*].
- Step-009: The Hosting CSE should respond with reasonable parameters in Response message for <*subscription*> Create operation.
- Step-010: The other AE that is Remote Controller requests to the Hosting CSE for discovery of [washer] resource with filter criteria.
- Step-011: The Hosting CSE checks its authority, and generates the results that meet the filter criteria.
- Step-012: The Hosting CSE should respond with reasonable parameters including Content in Response message for Retrieve operation.
- Step-013: The Remote Controller requests to update a particular attribute of [washer] resource with the ultimate purpose of controlling the physical device.
- Step-014: Depending on the result of privilege check, the Hosting CSE performs the requested operation.
- Step-015: According to the <subscription> created in the Step-008, the Hosting CSE should send a Notification to the Washer (AE).

- Step-016: The Washer should physically operate functions. In this example, it turn on the device if the value of powerStatus attribute changed from zero to one.
- Step-017: The Washer should respond with reasonable parameters in Response message indicating its result and status. (e.g. Success or Failure/On or Off).
- Step-018: The Hosting CSE should respond with reasonable parameters in Response message for Update operation.
- Step-019: The Remote Controller can see the success of failure and current status.

NOTE 2: In many other use cases, the remote controller (AE) can just retrieve the [*washer*] resource and its attributes. In other word, the [*washer*] resource may be used only for mornitoring.

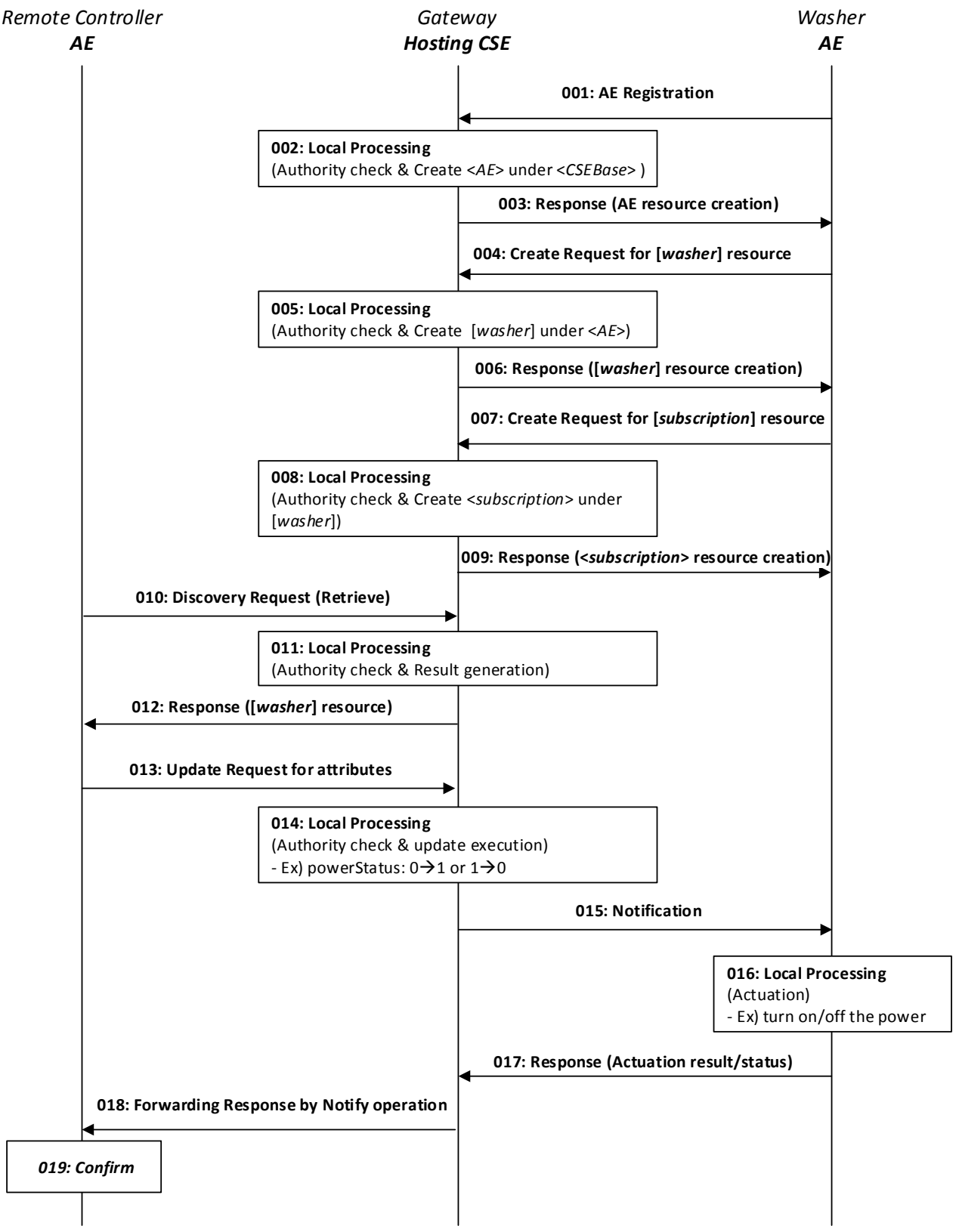

Figure 6.2.1.1-3: An example of [*washer*] procedure

6.2.1.2 two-level <*appliance*>

The first approach described in clause 6.2.1.1 is straightforward to implement. But, it is difficult to represent the relationships among services and shared attributes should be re-defined for each specializations everytime.

This clause describes the second approach introducing a new resource type <*applianceService*> which will be also specialized to specific services such as [*timer*], [*spin*], etc.

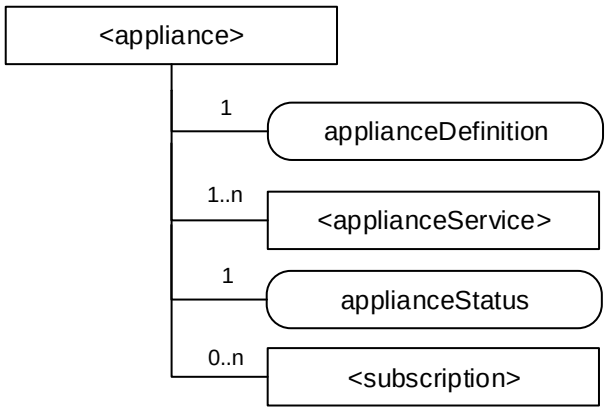

Figure 6.2.1.2-1: Structure of <*appliance*> resource

The <*appliance*> resource should contain the child resources specified in table 6.2.1.2-1.

Table 6.2.1.2-1: Child resources of <*appliance*> resource

Child Resources of <*appliance*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*applianceService*>	1..n	This resource provides capability to get information about common, shared or device-specific operational states of an appliance and to control an appliance with a specific set of commands linked to the appliance	<*appFuncAnnc*>
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The <*appliance*> resource should contain the attributes specified in table 6.2.1.2-2.

Table 6.2.1.2-2: Attribute of <*appliance*> resource

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description	<*applianceAnnc*> Attributes
resourceType	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
resourceID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
resourceName	1	WO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
parentID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
expirationTime	1	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
labels	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
creationTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
lastModifiedTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announceTo	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announcedAttribute	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
applianceStatus	1	RO	Status information of the application which is represented as the parent <*AE*> resource (e.g. online, offline)	OA

The <*applianceService*> is a set of characteristics supporting a service and it is re-usable for multiple specializations of <*appliance*> resource.

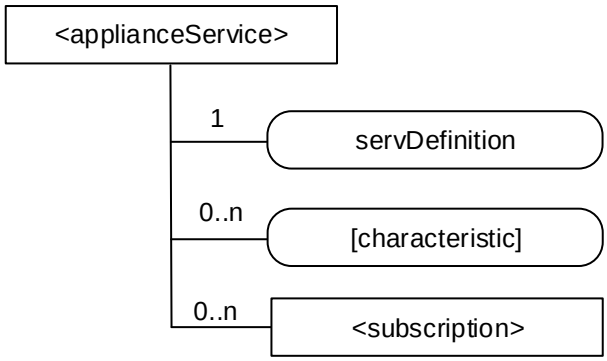

Figure 6.2.1.2-2: Structure of <*applianceService*> resource

The <*applianceService*> resource should contain the child resources specified in table 6.2.1.2-3.

Table 6.2.1.2-3: Child resources of <*applianceService*> resource

Child Resources of <*appliance*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The <*appliance*> resource should contain the attributes specified in table 6.2.1.2-4.

Table 6.2.1.2-4: Attribute of <*applianceService*> resource

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description	<*applianceAnnc*> Attributes
resourceType	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
resourceID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
resourceName	1	WO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
parentID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
expirationTime	1	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
labels	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
creationTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
lastModifiedTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announceTo	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announcedAttribute	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
servDefinition	1	WO	Definition of the service that is exposed by this resource (e.g. Common, Timer)	MA
[characteristic]	0..n	RW	List of attributes which are defined per each specialized <*applianceService*> resource. Specialized attributes can have characteristics or behaviors that are defined in oneM2M information model	OA

The <*applianceService*> resource should be specialized to particular services. The <*appliance*> resource shall have at least one specialization of the <*applianceService*> resource for indicating its common properties.

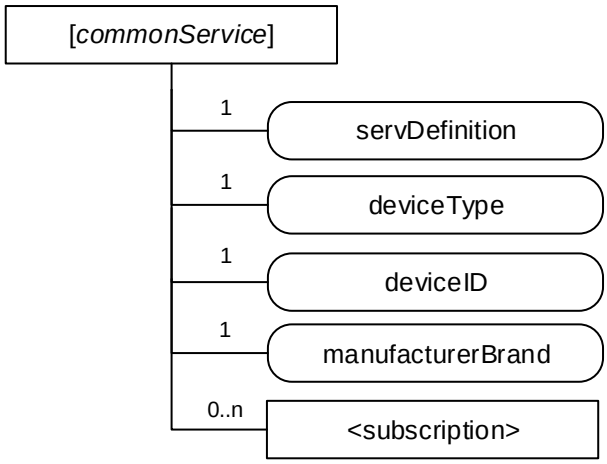

Figure 6.2.1.2-3: Structure of [*commonService*] resource

The [*commonService*] specialization of the <*applianceService*> resource should contain the child resources specified in table 6.2.1.2-5.

Table 6.2.1.2-5: Child resources of *[commonService]* specialization

Child Resources of <*appliance*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The [*commonService*] specialization should contain the attributes specified in table 6.2.1.2-6.

Table 6.2.1.2-6 Attribute of *[commonService]* specialization

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description	<*applianceAnnc*> Attributes
resourceType	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
resourceID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
resourceName	1	WO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
parentID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
expirationTime	1	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
labels	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
creationTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
lastModifiedTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announceTo	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announcedAttribute	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
servDefinition	1	WO	Definition of the application that is exposed by this resource (e.g. Washer)	MA
deviceType	1	RO	See clause 5.3.2	OA
deviceID	1	RO	See clause 5.3.2	OA
manufacturerBrand	1	RO	See clause 5.3.2	OA

The <*appliance*> resource should have more specializations of the <*applianceService*> resource for indicating its characteristics and behaviors.

EXAMPLE: It can have a [*timer*] resource.

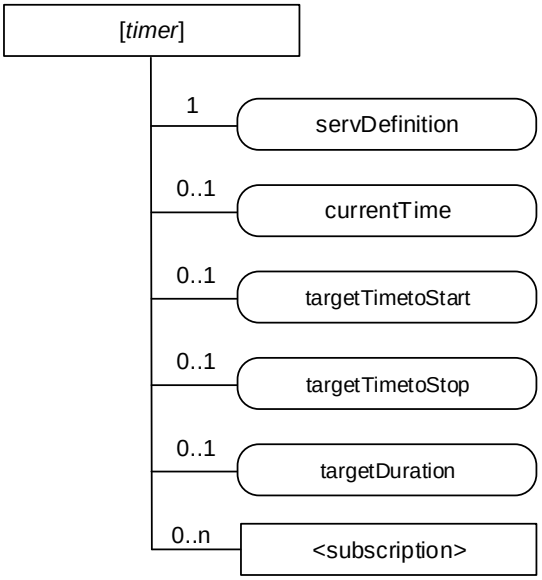

Figure 6.2.1.2-4: Structure of [*timer*] resource

The [*timer*] specialization of the <*applianceService*> resource should contain the child resources specified in table 6.2.1.2-7.

Table 6.2.1.2-7: Child resources of *[timer]* specialization

Child Resources of <*appliance*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The [t*imer*] specialization should contain the attributes specified in table 6.2.1.2-8.

Table 6.2.1.2-8: Attribute of *[timer]* specialization

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description	<*applianceAnnc*> Attributes
resourceType	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
resourceID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
resourceName	1	WO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
parentID	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
expirationTime	1	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
labels	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	MA
creationTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
lastModifiedTime	1	RO	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announceTo	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
announcedAttribute	0..1 (L)	RW	See clause 9.6.1.3 in oneM2M TS-0001 [i.19]	NA
currentTime	0..1	RW	Current time	OA
targetTimetoStart	0..1	RW	The time when the device is expected to start its operation	OA
targetTimetoStop	0..1	RW	The time when the device is expected to stop its operation	OA
targetDuration	0..1	RW	Target duration of the operation	OA

Example B. Resource *washer* (Specialization of two-level <*appliance*>)

The [*washer*] resource is used to share information regarding wahser. The [*washer*] is a specialization of the two-level <*appliance*> resource.

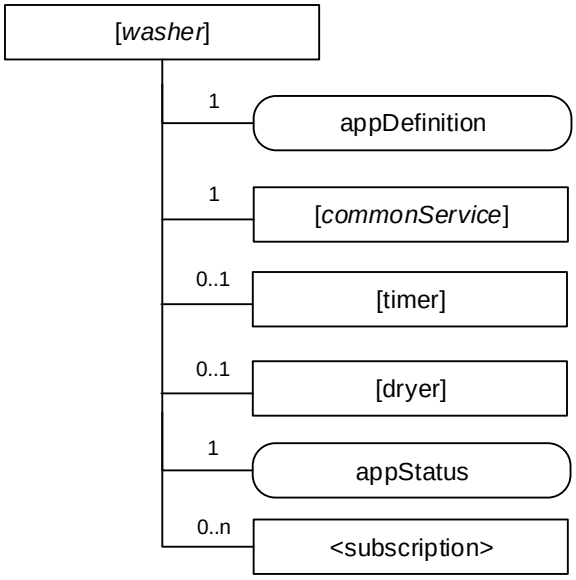

Figure 6.2.1.2-5: Structure of [*washer*] resource

The [*washer*] resource should contain the child resource specified in table 6.2.1.2-9.

Table 6.2.1.2-9: Child resources of [*washer*] resource

Child Resources of <*washer*>	Child Resource Type	Multiplicity	Description	<*applianceAnnc*> Child Resource Types
commonService	<*applianceService*> *as defined in the specialization of [commonService] (common service)*	1	See figure 6.2.1.2-3	<*applianceServiceAnnc*>
[variable]	<*applianceService*> *as defined in the specialization of [timer] (shared service)*	0..1	See figure 6.2.1.2-4	<*applianceServiceAnnc*>
[variable]	<*applianceService*> *as defined in the specialization of [dryer] (device specific service)*	0..1	This resource describes a dryer function for a washer.	<*applianceServiceAnnc*>
[variable]	<*subscription*>	0..n	See clause 9.6.8 in oneM2M TS-0001 [i.19]	<*subscription*>

The <*washer*> resource should contain the attributes specified in table 6.2.1.2-10.

Table 6.2.1.2-10: Attribute of <*washer*> resource

Attributes of <*appliance*>	Multiplicity	RW/ RO/ WO	Description
resourceType	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
resourceID	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
resourceName	1	WO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
parentID	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
expirationTime	1	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
labels	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
creationTime	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
lastModifiedTime	1	RO	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
announceTo	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]
announcedAttribute	0..1 (L)	RW	See clause 9.6.2.3. in oneM2M TS-0001 [i.19]

Attributes of *<appliance>*	Multiplicity	RW/ RO/ WO	Description
applianceDefinition	1	WO	Definition of the application that is exposed by this resource (e.g. Washer)
applianceStatus	1	RO	Status information of the application which is represented as the parent <*AE*> resource (e.g. online, offline)

6.3 Approach 2: Existing Resource Type container/contentInstance

6.3.1 Introduction

This clause introduces another candidate solution for respresentation of the oneM2M Information Model defined in the clause 5.3.2 using container and contentInstance.

With <container> resource, it is also possible to map the home appliance information model to oneM2M system.

Both <appliance> and <appService> resources described in Approach 1 can be converted into <container> respectively. Because <container> resource type supports hierarchical structure, it can have a <container> as a child resource. The parent <container> is mapped with a specific device (appliance), and the child <container> is mapped with a specific service (appService). Attributes that is located directly under the root (applianceDefinition, applianceStatus) are also mapped to <container>.

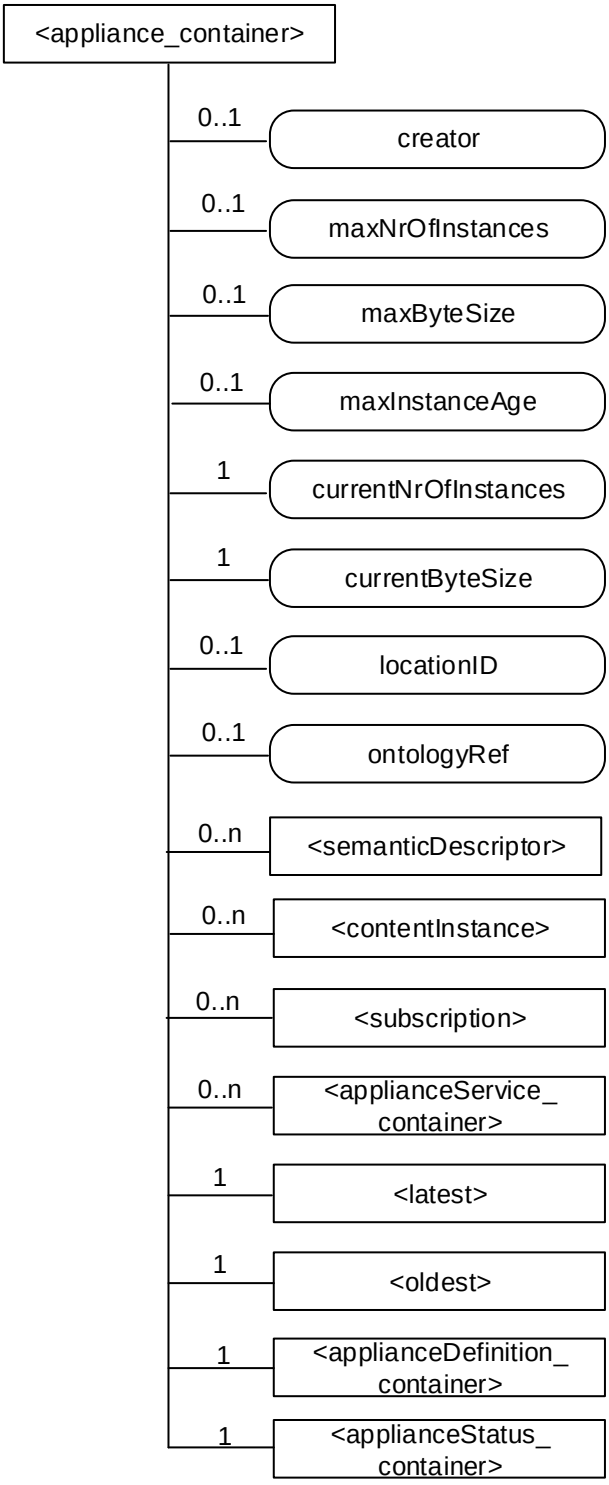

Figure 6.3.1-1: Structure of <*container*> resource for representation of appliance

Each services can be represented using <*container*> as well and characteristics for the services can be also represented as <*container*>.

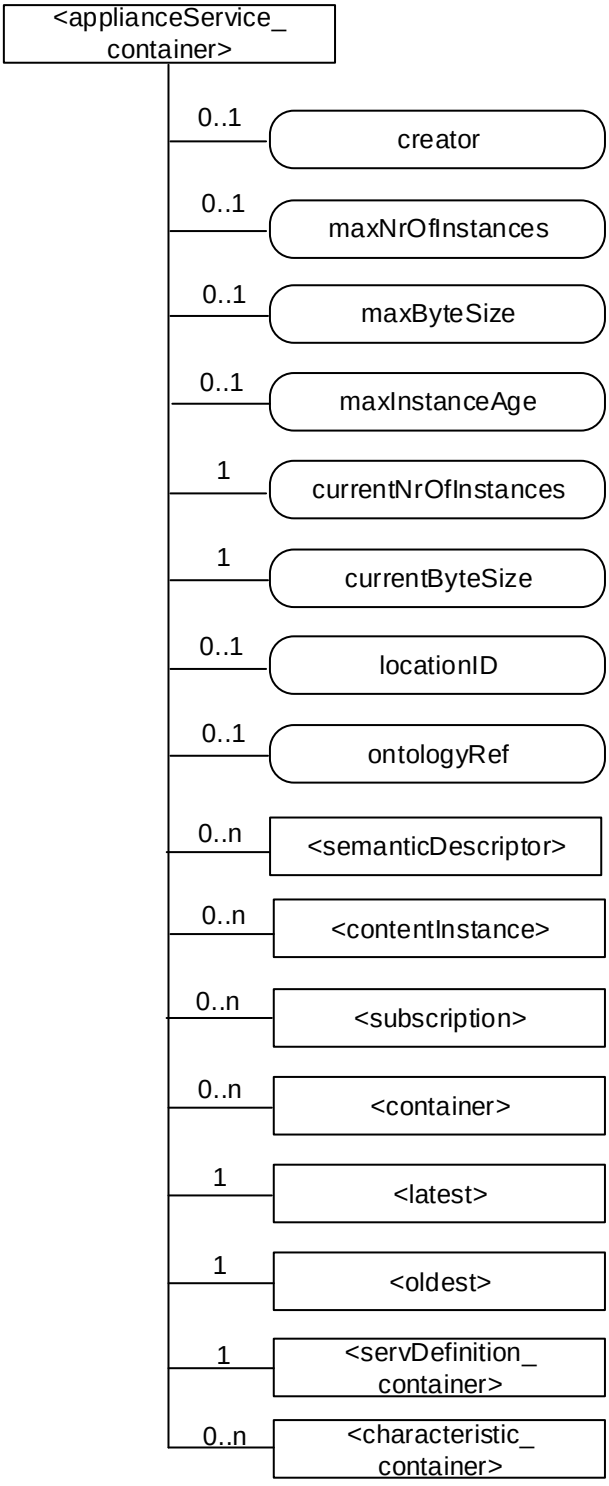

Figure 6.3.1-2: Structure of a child <container> resource for representation of appService

6.4 Comparison of potential solutions

Table 6.4-1

	Using new resource type (appliance)	Using existing resource type (container)
Advantages	• Less overhead • More efficient for use of storage • Special features for home appliances can be defined (e.g. application triggering between different appliances) • Subscription for specific domain • True abstraction (application doesn't need to understand technology specific data model) • Application developers will find the new resource type easier to use	• More flexible for usage by applications (coarse grain container or hierarchical container) • Applicable for more general purposes
Disadvantages	• Maintainability • Increases the complexity of system • Reduces the interoperability	• Ontologies needed to make it understandable • Mandatory attributes for container cause overhead • Difficult to monitor directly specific attributes (coarse grain container) or heavier overhead (hierarchical container)

7 Conclusions

In oneM2M system it is necessary to develop the common way of controlling and monitoring Home Appliances regardless of device manufacturers. For that reason an abstract information model for Home Appliances can be used. The information model not only guarantees cross-vendor interoperability, but also facilitates machine-to-machine interaction without human's interaction. The present document "Home Domain Abstract Information Model" introduces several approaches for defining the information model and how to map the proposed information model into resource in oneM2M system.

After having studied about external technologies such as AllJoyn, SDT, OIC and ECHONET, several approaches for defining information model are suggested and some information model examples based on the approaches are brought (clause 5). While studying on these approaches, it was also proposed to apply SDT 3.0 as it is for the modeling efforts because it is alreay well designed to be generic and reviewed by many technology providers. Thus, it is recommended to design the oneM2M device information models for home appliances based on SDT 3.0 with utilization of exercises done in the present document.

The information model to be defined should be represented as resource in oneM2M. It can be done using resource types that allow sharing data between applications. Advantages and disadvantages of using them are shown in a table (clause 6). This can be used as a guideline for the future normative work.

The present document will lead to normative work (oneM2M TS-0023 [i.18]) that will contain more specific information models for Home Appliances such as air conditioner, refrigerator, oven, washing maching and robot cleaner and to CRs (oneM2M TS-0001 [i.19]) that will reflect changes to map the information model into resource.

History

Publication history		
V2.0.0	30-Aug-2016	Publication