|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* |  |
| Source:\* |  |
| Date:\* |  |
| Reason for Change/s:\* |  |
| CR against: Release\* |  |
| CR against: WI\* | Active <#WI#>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number:  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* |  |
| Clauses \* |  |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Other TS/TR(s) impacted | None |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2019 (do not modify) | |

**oneM2M Notice**

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

GUIDELINES for Change Requests:

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

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

In case of a correction, and the change apply to previous releases, a separate “mirror CR” should be posted at the same time of this CR

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

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

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

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

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

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

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

## Introduction

This is the first conversion of TR-0039 to Markdown format.

<https://git.onem2m.org/specifications/tr-0039/-/merge_requests/2>

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

---a/TR-0039-Developer\_guide-SDT-based\_implementation.md  
+++b/TR-0039-Developer\_guide-SDT-based\_implementation.md

@@ -1,41 +1,34 @@

![oneM2M logo](media/logo.png)

![](media/image1.png)

\*\*oneM2M Technical Report\*\*

|oneM2M<br />Technical Report |oneM2M<br />Technical Report |

|||

|-|-|

|Document Number |TR-00xx-V-x.y.z |

|Document Name: |The name of the document<br /> |

|Date: |&lt;20yy-mm-dd> |

|Abstract: |&lt; An abstract of the document and information that may be used in subsequent electronic searches> |

|Template Version: January 2020 (do not modify) | |

|Document Number |TR-0039-V-2.0.0 |

|Document Name: |Developer guide: Interworking Proxy using SDT |

|Date: |2018-03-12 |

|Abstract: |The document describes how a developer can easily implement interworking with non-oneM2M devices using the SDT (Smart Device Template) defined in TS-0023 (Home appliances Information Model). |

|Template Version: January 2017 (Do not modify) |Template Version: January 2017 (Do not modify) |

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

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

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

About oneM2M

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

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

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

Copyright Notification

(c) 2020, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC).

(c) 2018, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC).

All rights reserved.

The copyright and the foregoing restriction extend to reproduction in all media.

Notice of Disclaimer & Limitation of Liability

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

@@ -46,175 +39,1213 @@ NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURA

# Contents

# 1 Scope

The present document ...

The present document describes how a developer can quickly and easily implement interworking with non-oneM2M devices using the SDT (Smart Device Template) defined in TS-0023 <a href="#\_ref\_i.2">[i.2]</a> (Home appliances Information Model). As an example, a scenario of an application controlling and monitoring a connected lamp, already commercially available on the market, is proposed. The goal is to describe with an example the methodology for building this interworking, allowing the developers to apply it to any other non-oneM2M devices using the abstraction layer provided by SDT.

`EXAMPLE: The present document provides the necessary adaptions to the endorsed document.`

<mark>The Scope shall not contain requirements.</mark>

To focus on the topics described above, the security aspect is not considered in the scope of this developer guide, especially Access Control Policy issue is not discussed

# 2 References

<mark>The following text block applies.</mark>

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

## 2.1 Normative references

<mark>As a Technical Report (TR) is entirely informative it shall not list normative references.</mark>

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

Normative references are not applicable in the present document.

Not applicable.

<mark>Editor note: There are references to [1] which is assumed is TS-0023 -> i.2</mark>

## 2.2 Informative references

<mark>Clause 2.2 shall only contain informative references which are cited in the document itself.</mark>

References are either specific (identified by date of publication and/or edition number or version number) or nonspecific. 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.

- <a name="\_ref\_i.1">[i.1]</a> oneM2M Drafting Rules [https://member.onem2m.org/static\_Pages/others/Rules\_Pages/oneM2M-Drafting-Rules-V1%202%202.doc](https://member.onem2m.org/static\_Pages/others/Rules\_Pages/oneM2M-Drafting-Rules-V1%202%202.doc)

- <a name="\_ref\_i.1">[i.1]</a> oneM2M Drafting Rules.

> NOTE: Available at [http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf](http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf).

- <a name="\_ref\_i.2">[i.2]</a> oneM2M TS-0023: "Home Appliances Information Model and Mapping".<mark>Editor note: Change title</mark>

- <a name="\_ref\_i.3">[i.3]</a> oneM2M TS-0001: "Functional Architecture".

- <a name="\_ref\_i.4">[i.4]</a> oneM2M TS-0004: "Service Layer Core protocol Specification".

- <a name="\_ref\_i.5">[i.5]</a> oneM2M TS-0009: "HTTP Protocol Binding".

# 3 Definition of terms, symbols and abbreviations

<mark>Delete from the above heading the word(s) which is/are not applicable.</mark>

# 3 Abbreviations

## 3.1 Terms

<mark>Clause numbering depends on applicability.</mark>

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

`ADN-AE AE which resides in the Application Dedicated Node`

`AE Application Entity`

`CSE Common Services Entity`

`CSE-ID Common Service Entity Identifier`

`IN Infrastructure Node`

`IN-AE Application Entity that is registered with the CSE in the Infrastructure Node`

`IN-CSE CSE which resides in the Infrastructure Node`

`MN-AE Application Entity that is registered with the CSE in Middle Node`

`MN-CSE CSE which resides in the Middle Node`

`URI Uniform Resource Identifier`

- <mark>A definition shall not take the form of, or contain, a requirement.</mark>

- <mark>The form of a definition shall be such that it can replace the term in context. Additional information shall be given only in the form of examples or notes (see below).</mark>

- <mark>The terms and definitions shall be presented in alphabetical order.</mark>

For the purposes of the present document, the [following] terms and definitions [given in ... and the following] apply:

# 4 Conventions

<mark>Definition format</mark>

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 <a href="#\_ref\_i.1">[i.1]</a>.

<mark>&lt;defined term>: &lt;definition></mark>

<mark>If a definition is taken from an external source, use the format below where [N] identifies the external document which must be listed in Section 2 References.</mark>

# 5 Use Case

<mark>&lt;defined term>[N]: &lt;definition></mark>

## 5.0 Introduction

<mark>example 1: text used to clarify abstract rules by applying them literally</mark>

This clause describes a use case which helps to understand the devices abstraction layer concept of oneM2M through the combined use of the SDT-based (Smart Device Template) Information Model specified in oneM2M TS-0023 <a href="#\_ref\_i.2">[i.2]</a> and the Interworking Proxy Entity.

> NOTE: This may contain additional information.

## 3.2 Symbols

## 5.1 Abstraction description via the lightbulbs example

<mark>Clause numbering depends on applicability.</mark>

This guide is based on the following use case: an application on a Gateway, Cloud/Server, or Smartphone wants to monitor and control lightbulbs. Due to the abstraction layer provided by oneM2M, the monitoring and controlling processes can be performed independently from underlying connectivity technologies. This tutorial is based on the example of Philips Hue lightbulbs (these lightbulbs require dedicated bridge for connection with Gateway). Figure 5.1-1 presents the general architecture of the use case.

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

NOTE: The terms of use of Philips Hue API are available at https://developers.meethue.com/documentation/terms-use.<mark>Editor note: check this URL</mark>

<mark>Symbol format</mark>

![LightbulbsUseCase](media/LightbulbsUseCase.png)

<mark>`&lt;symbol> &lt;Explanation>`</mark>

<mark>`&lt;2nd symbol> &lt;2nd Explanation>`</mark>

<mark>`&lt;3rd symbol> &lt;3rd Explanation>`</mark>

\*\*Figure 5.1-1: Overview of the lightbulbs use case\*\*

## 3.3 Abbreviations

The main components are introduced as follows:

<mark>Abbreviations should be ordered alphabetically.</mark>

- The lightbulbs are deployed in a home and are attached to a home gateway via Philips Hue Bridge.

- The Gateway communicates with a Cloud/Server platform allowing the lightbulbs to be controlled remotely by the Smartphone.

- The Cloud/Server platform and Gateway supports a set of services to enable the Smartphone to more easily control and monitor the lightbulbs in the home. Some examples of services include device abstraction layer, registration, discovery, data management, group management, subscription/notification, etc.

- The Smartphone hosts an application used to remotely control and monitor the lightbulbs in the home and supports the following capabilities:

- Discovery of the lightbulbs deployed in the home.

- Sending commands to change light state i.e. switch on/off and change colour.

- Receive state of a lightbulb.<!---->

<mark>Clause numbering depends on applicability.</mark>

# 6 Introduction to IPE and SDT

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

## 6.1 Introduction to IPE

<mark>Abbreviation format</mark>

An IPE (Interworking Proxy Entity) is a specialized AE (Application Entity) that allows the oneM2M system to interact with any non-oneM2M system, in a seamless way, through the Mca interface. It has the capability to remap the specific data model to oneM2M resources (&lt;AE>, &lt;container>, &lt;flexContainer>, etc.) and maintain bidirectional communication with the non-oneM2M system.

<mark>`&lt;ABBREVIATION1> &lt;Explanation>`</mark>

<mark>`&lt;ABBREVIATION2> &lt;Explanation>`</mark>

<mark>`&lt;ABBREVIATION3> &lt;Explanation>`</mark>

# 4 Conventions

## 6.2 Introduction to SDT

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 <a href="#\_ref\_i.1">[i.1]</a>

### 6.2.1 SDT data model description

The SDT (Smart Device Template) is a reference template to model most home appliance functions in a unified way which is a result of consensus amongst various SDOs and industry alliances. Abstraction from the various underlying home-area network technologies and getting an unified way of controlling/commanding the appliances are among the key goals of the SDT.

# 5 User defined clause(s) from here onwards

&lt;Text>

The SDT approach is to define re-usable basic functions (or services) (labelled "ModuleClass" in Figure 6.2-1) which can represent the typical functions found, for example, in many home automation systems, such as "on/off", "dim a lamp", "receive events from binary sensor", "read data from sensor", etc.

![SDTGenericDevice](media/SDTGenericDevice.png)

## 5.1 User defined subdivisions of clause(s) from here onwards

&lt;Text>

\*\*Figure 6.2.1-1: SmartHome Device Template for a generic device\*\*

<mark>The following text is to be used when appropriate:</mark>

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) 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.

The SDT is available under Apache License 2 at oneM2M's GitLab: [https://git.onem2m.org/MAS/SDT](https://git.onem2m.org/MAS/SDT)

# <mark>Proforma copyright release text block</mark>

<mark>This text box shall immediately follow after the heading of an element (i.e. clause or annex) containing a proforma or template which is intended to be copied by the user. Such an element shall always start on a new page.</mark>

<mark>Editor note: Add informative reference to SDT</mark>

<mark>Notwithstanding the provisions of the copyright clause related to the text of the present document, oneM2M grants that users of the present document may freely reproduce the &lt;proformatype> proforma in this {clause|annex} so that it can be used for its intended purposes and may further publish the completed &lt;proformatype>.</mark>

<mark>&lt;PAGE BREAK></mark>

### 6.2.2 SDT Device example: deviceLight

A light is a device that is used to control the state of an illumination device. This Device has one mandatory \*binarySwitch\* Module and the following optional Modules: \*faultDetection\*, \*runState\*, \*colour\*, \*colourSaturation\*, \*brightness\*.

## <mark>Annexes</mark>

<mark>Each annex shall start on a new page (insert a page break between annexes A and B, annexes B and C, etc.).<mark>

<mark>Editor note: Remove references in table names. Add a generic reference to TS-0023</mark>

# Annex &lt;A>:<br />Title of annex

\*\*Table 6.2.2\*\* \*\*-1: Modules of\*\* \*\*deviceLight\*\* \*\*Device model\*\* \*\*(from <a href="#\_ref\_1">[1]</a>)\*\*

&lt;Text>

|Module Instance Name |Module Class Name |Optional |Description |

|-|-|-|-|

|faultDetection |faultDetection |true |See clause 5.3.16 |

|binarySwitch |binarySwitch |false |See clause 5.3.5 |

|runState |runState |true |See clause 5.3.28 |

|colour |colour |true |See clause 5.3.10 |

|colourSaturation |colourSaturation |true |See clause 5.3.11 |

|brightness |brightness |true |See clause 5.3.8 |

<mark>&lt;PAGE BREAK></mark>

The faultDetection ModuleClass provides information about whether a fault has occurred in the actual device.

# Annex &lt;B>:<br />Title of annex

\*\*Table 6.2.2-2: DataPoints of faultDetection ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

&lt;Text>

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|status |xs:boolean |true |false |false |Status of fault detection. |

|code |xs:integer |true |false |true |Code of the fault. |

|description |xs:string |true |false |true |Message of the fault. |

## <!--B.1--> First clause of the annex

The binarySwitch ModuleClass provides capabilities to control and monitor the state of power.

&lt;Text>

\*\*Table 6.2.2-3: Actions of binarySwitch ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

### B.1.1 First subdivided clause of the annex

|Return Type |Name |Argument |Optional |Documentation |

|-|-|-|-|-|

|none |toggle |none |true |Toggle the switch. |

&lt;Text>

<mark>&lt;PAGE BREAK></mark>

\*\*Table 6.2.2-4: DataPoints of binarySwitch ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

# Annex &lt;y>:<br />Bibliography

<mark>The annex entitled "Bibliography" is optional.</mark>

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|powerState |xs:boolean |true |true |false |The current status of the \*binarySwitch\*. "True" indicates turned-on, and "False" indicates turned-off. |

<mark>It shall contain a list of standards, books, articles, or other sources on a particular subject which are not mentioned in the document itself.</mark>

The runState ModuleClass provides capabilities to control and monitor machine state of appliances.

<mark>It shall not include references mentioned in the document.</mark>

\*\*Table 6.2.2-5: DataPoints of runState ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|currentMachineState |hd:machineState |true |true |false |Currently active machine state. |

|machineStates |list of hd:machineState |true |false |false |List of possible machine states the device supports (see clause 5.5.15) |

|progressPercentage |float |true |false |true |Indication of current progress in percentage |

- <mark>&lt;Publication>: "&lt;Title>".</mark>

The \*colour\* ModuleClass provides the capabilities to set the value of <mark>Red, Green, Blue for the colour device</mark>.

<mark>OR</mark>

\*\*Table 6.2.2-6: DataPoints of colour ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

<mark>&lt;Publication>: "&lt;Title>".</mark>

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|red |xs:integer |true |true |false |The R value of RGB; the range is [0,255] |

|green |xs:integer |true |true |false |The G value of RGB; the range is [0,255] |

|blue |xs:integer |true |true |false |The B value of RGB; the range is [0,255] |

<mark>&lt;PAGE BREAK></mark>

The \*colourSaturation\* ModuleClass describes a colour saturation value. The value is an integer. A \*colourSaturation\* has a range of [0,100]. A \*colourSaturation\* value of 0 means producing black and white images. A \*colourSaturation\* value of 50 means producing device specific normal colour images. A \*colourSaturation\* value of 100 means producing device very colourfull images.

\*\*Table 6.2.2-7: DataPoints of colourSaturation ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

# History

<mark>This clause shall be the last one in the document and list the main phases (all additional information will be removed at the publication stage).</mark>

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|colourSaturation |xs:integer |true |true |false |The status of colour saturation level. |

The \*brightness\* ModuleClass describes the brightness of a light e.g. from a lamp. Brightness is scaled as a percentage. A lamp or a monitor can be adjusted to a level of light between very dim (0 % is the minimum brightness) and very bright (100 % is the maximum brightness).

|Publication history |Publication history |Publication history |

|-|-|-|

|V1.1.1 |&lt;yyyy-mm-dd> |&lt;Milestone> |

| | | |

| | | |

| | | |

| | | |

\*\*Table 6.2.2-8: DataPoints of brightness ModuleClass (from <a href="#\_ref\_1">[1]</a>)\*\*

|Name |Type |Readable |Writable |Optional |Documentation |

|-|-|-|-|-|-|

|brightness |xs:integer |true |true |false |The status of brightness level in percentage. |

## 6.3 IPE and SDT in oneM2M tree

In reference to oneM2M terminology, when a new device is discovered the CSE should register it and map its SDT-based representation to oneM2M common resources according to the rules defined in TS-0023 <a href="#\_ref\_i.2">[i.2]</a>.

To allow current non-oneM2M devices ("NoDN", Non-oneM2M Device Node) to connect to the oneM2M system the specification provides "Interworking/Integration of non-oneM2M solutions and protocols" section (Annex F of oneM2M TS-0023 <a href="#\_ref\_i.2">[i.2]</a>). Figure 6.3-1 shows SDT in oneM2M architecture.

![DataModelTranslation](media/DataModelTranslation.png)

\*\*Figure 6.3-1: Translation of non-oneM2M Data Model to SDT Data Model via the IPE\*\*

# 7 Functional architecture

Clause 7 describes how the elements of this use case are represented by corresponding oneM2M architectural entities. Figure 7-1 presents the functional architecture of this use case.

![FunctionalArchitecture](media/FunctionalArchitecture.png)

\*\*Figure 7-1: oneM2M functional architecture of the bulbs use case\*\*

There are two sections in the functional architecture of this use case. On the left side of the red dashed line there is the non-oneM2M section based on Philips Hue architecture. It consists of two Hue Bulbs and Hue Bridge.

On the right side there is the oneM2M architecture section which consists of MN-CSE (Middle Node - Common Service Entity) and AE: IPE (Application Entity: Interworking Proxy Entity) on the Gateway, IN-CSE (Infrastructure Node CSE) on the Cloud/Server, ADN-AE-1 (Application Dedicated Node) on the Smartphone and Mcc and Mca reference points.

Mca reference point is used to interface an AE and a CSE together. Mcc reference point is used to interface CSEs. In this use case Mca is used between ADN-AE-1 on the Smartphone and IN-CSE on the Cloud/Server, while Mcc is used between MN-CSE on the Gateway and IN-CSE on the Cloud/Server.

AE: IPE makes non-oneM2M devices visible to oneM2M platform, so it allows ADN-AE-1 on the Smartphone to remotely control and monitor the bulbs. AE: IPE mechanism is described in details in clause 6.3.

# 8 Procedures and call flows

## 8.1 Introduction

The deployment of the oneM2M standard in the present use case requires procedures that are classified as follows:

- \*\*Registration:\*\* AE registration with CSE: IPE-AE with MN-CSE, ADN-AE with IN-CSE.

- \*\*Resources creation:\*\* a set of \_&lt;flexContainer>\_ resources creation to represent SDT Device in a resource tree.

- \*\*Discovery of resources with filter criteria:\*\* discovery of resources with given containerDefinition attribute filtered with filter criteria to receive associated URI.

- \*\*Discovery of children resources:\*\* discovery of a resource tree associated with particular SDT Device to get access to particular ModuleClass, DataPoint, Action.

- \*\*Get resource\*\* : get value of DataPoints through sending RETRIEVE request.

- \*\*Change resource:\*\* changing a DataPoint value or triggering an Action through sending UPDATE request.

- \*\*Subscription and notification mechanism\*\* : monitor DataPoints values changes through subscription and notification mechanism.

In this developer guide there is an assumption that MN-CSE is registered with IN-CSE. More assumptions are listed in clause 9.2.

## 8.2 Call Flows from perspective of device adapter developer

### 8.2.1 IPE-AE registration with MN-CSE

The procedure to register device adapter (IPE-AE) with gateway (MN-CSE) is as follows:

- Device adapter (IPE-AE) sends registration request to Gateway (MN-CSE).

- Gateway (MN-CSE) responds with status of the registration and Content-Location header for the registered entity.

![AERegistration](media/AERegistration.png)

\*\*Figure 8.2.1-1: AE registration with CSE request\*\*

### 8.2.2 SDT Device resource tree creation in MN-CSE

The procedure to create resource tree for particular SDT Device (deviceLight in this use case) is as follows:

- Device adapter (IPE-AE) sends a CREATE request to gateway (MN-CSE) to create \_&lt;flexContainer>\_ for SDT Device.

- Gateway (MN-CSE) responds with URI of the new \_&lt;flexContainer>\_ for SDT Device.

- Device adapter (IPE-AE) sends a CREATE reqest to gateway (MN-CSE) to create a \_&lt;flexContainer>\_ for a single Module with associated customAttributes (DataPoints). Please note that there is sent one such request per each Module.

- Gateway (MN-CSE) responds with URI for created Module.

- Device adapter (IPE-AE) sends a CREATE request to gateway (MN-CSE) to create \_&lt;flexContainer>\_ for a single Action. The \_&lt;flexContainer>\_ which represents Action is set as a child of \_&lt;flexContainer>\_ which represents Module.

- Gateway (MN-CSE) responds with URI for created Action.

![ResourceTreeSDTDevice](media/ResourceTreeSDTDevice.png)

\*\*Figure 8.2.2-1: Creation of resource tree for SDT Device\*\*

## 8.3 Call flows from perspective of utility application developer

### 8.3.1 Application Entity registration in IN/MN-CSE

The procedure to register utility application (ADN-AE-1) with server (IN-CSE) is as follows:

- Utility application (ADN-AE-1) sends registration request to gateway (IN-CSE).

- Server (IN-CSE) responds with status of the registration and Content-Location header for the registered entity.

![ADNAERegistration](media/ADNAERegistration.png)

\*\*Figure 8.3.1-1: AE registration with CSE request\*\*

### 8.3.2 Discovery Requests

The procedure to discover SDT Device is as follows:

- Utility application (ADN-AE-1) sends a RETRIEVE request to server (IN-CSE) including filter criteria conditions, especially it could be a \_containerDefinition\_ attribute (e.g. org.onem2m.home.devices.deviceLight).

- Server (IN-CSE) responds with list of URIs of the discovered resources, if any (especially URI for deviceLight which represents a bulb).

![DiscoverySDTDevice](media/DiscoverySDTDevice.png)

\*\*Figure 8.3.2-1: Discovery of SDT Device with filter criteria\*\*

### 8.3.3 Control & monitor devices

### 8.3.3.1 Getting URLs for resources

The procedure to retrieve needed URIs for Modules which are associated with particular Device is as follows:

- Utility application (ADN-AE-1) sends a request to server (IN-CSE) with URI of the Device.

- Server (IN-CSE) responds with a list of the Modules URIs.

![DiscoveryModules](media/DiscoveryModules.png)

\*\*Figure 8.3.3.1-1: Discovery of Modules related to SDT Device\*\*

### 8.3.3.2 DataPoint value retrieving

Retrieving value of a DataPoint is possible only through retrieving all DataPoints which belong to particular ModuleClass.

The procedure to get values of all DataPoints which belong to particular ModuleClass is as follows:

- Utility application (ADN-AE-1) sends a RETRIEVE request to server (IN-CSE) with information which ModuleClass's DataPoints should be retrieved.

- Server (IN-CSE) responds with current values of all DataPoints which belong to the ModuleClass that was requested.

![DataPointRetrieve](media/DataPointRetrieve.png)

\*\*Figure 8.3.3.2-1: DataPoint retrievement\*\*

### 8.3.3.3 DataPoint value changing

The procedure to change DataPoint value is as follows:

- Utility application (ADN-AE-1) sends an UPDATE request to server (IN-CSE) with \_&lt;flexContainer>\_ which contains the new value of the DataPoint. There is also a possibility to change more than one DataPoint of particular ModuleClass in a single request (in this case the &lt;flexContainer> contains new values of all needed DataPoints.

- Server (IN-CSE) responds with status of DataPoint's UPDATE operation.

![DataPointValueChange](media/DataPointValueChange.png)

\*\*Figure 8.3.3.3-1: Change value of a DataPoint\*\*

### 8.3.3.4 Action triggering

The procedure to trigger an Action with no arguments is as follows:

- Utility application (ADN-AE-1) sends an UPDATE request to server (IN-CSE) with empty \_&lt;flexContainer>\_.

- Server (IN-CSE) responds with status of UPDATE operation.

![ActionTriggering](media/ActionTriggering.png)

\*\*Figure 8.3.3.4-1: Action triggering procedure\*\*

### 8.3.3.5 Subscription mechanism

Following diagrams show four parts of subscription mechanism: subscription creation, listener's status verification, notification message triggered by DataPoint value change and subsription deletion.

The procedure of creating a \_&lt;subscription>\_ resource is as follows:

- Utility application (ADN-AE-1) sends a CREATE request to server (IN-CSE).

- Server (IN-CSE) responds with all parameters of created \_&lt;subscription>\_ resource.

![SubscriptionCreation](media/SubscriptionCreation.png)

\*\*Figure 8.3.3.5-1: Subscription creation procedure\*\*

There's a mechanism that allows Server to check listener's availability. The procedure of listener's (ADN-AE-1) status verification is as follows:

- Server (IN-CSE) sends verification request to listener (ADN-AE-1).

- Listener (ADN-AE-1) responds with HTTP 200 OK status.

![ListenerVerification](media/ListenerVerification.png)

\*\*Figure 8.3.3.5-2: Listener's status verification procedure\*\*

A change of DataPoint's value triggers a notification message with new DataPoint value that is sent to listener. The procedure of notification mechanism is as follows:

- DataPoint value change occurs.

- Server (IN-CSE) sends to utility application (ADN-AE-1) notification with new DataPoint value.

- Utility application (ADN-AE-1) sends confirmation of receiving the notification.

![NotificationMechanism](media/NotificationMechanism.png)

\*\*Figure 8.3.3.5-3: Notification mechanism procedure\*\*

The procedure of deletion of a &lt;subsription> resource is as follows:

- Utility application (ADN-AE-1) sends a DELETE request to server (IN-CSE).

- Server (IN-CSE) responds with deletion confirmation.

![SubscriptionDeletion](media/SubscriptionDeletion.png)

\*\*Figure 8.3.3.5-4: Deletion of &lt;subscription> resource procedure\*\*

# 9 Implementation

## 9.1 Introduction

Clause 9 provides implementation examples from two different perspectives: developer of a device adapter (AE: Inter-working Proxy) in clause 9.3 and developer of a utility application in clause 9.4.

![Figure 9.1-1: Implementation of TS-0023 <a href="#\_ref\_i.2">[i.2]</a> Module Classes for the Hue bulb use case](media/ImplementationOfTS-0023.png)

\*\*Figure 9.1-1: Implementation of TS-0023 <a href="#\_ref\_i.2">[i.2]</a> Module Classes for the Hue bulb use case\*\*

## 9.2 Assumptions

### 9.2.0 Introduction

- MN-CSE is registered with IN-CSE.

- Philips Hue Bridge and MN-CSE are on the same LAN.

- Philips Hue Bulbs are paired with the Philips Hue Bridge.

- Philips Hue Bridge is not directly represented in oneM2M resource tree, but through its associated IPE.

- Access Control Policy issue isn't discussed here because it's a security aspect which isn't considered in the scope of this developer guide.

|Draft history (to be removed on publication) |Draft history (to be removed on publication) |Draft history (to be removed on publication) |

### 9.2.1 Addressing for Entities

Each oneM2M entity including AE and CSE are addressable with correct host address that can be IP addresses or FQDN addresses resolved to IP addresses by DNS network services according to addressing rules specified in oneM2M standards.

The IN-CSE and MN-CSE entities presented in this use case are addressable with the following identifiers.

- IN-CSE:

- CSE-ID: /\*\*in-cse\*\*

- resourceName of IN-CSE's CSEBase resource: \*\*server\*\*

- IN-CSE FQDN\*\*:\*\* \*\*incse.provider.com\*\*

- IN-CSE HTTP port: \*\*8080\*\*

- MN-CSE:

- CSE-ID: /\*\*mn-cse\*\*

- resourceName of MN-CSE's CSEBase resource: \*\*home\\_gateway\*\*

- MN-CSE FQDN: \*\*mncse.provider.com\*\*

- MN-CSE HTTP port: \*\*8080\*\*

The host of Utility Application (ADN-AE-1) has following IP address: 192.168.10.1.

## 9.3 Developer of the device adapter (IPE-AE)

### 9.3.0 Introduction

This clause describes implementation process from the IPE-AE developer point of view.

![Figure 9.3.0-1: Scope of 9.3 Developer of the device adapter section](media/ScopeOfDeveloper.png)

\*\*Figure 9.3.0-1: Scope of 9.3 Developer of the device adapter section\*\*

### 9.3.1 IPE-AE registration with MN-CSE

The IPE-AE with MN-CSE registration is shown in the following procedure.

<mark>Editor note: RVI is missing in the requests</mark>

<mark>Editor note: Add \*\*HTTP Request\*\* and \*\*HTTP Response\*\* to the examples</mark>

``` http

POST /home\_gateway HTTP/1.1

Host: mncse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml;ty=2

X-M2M-RI: home\_gateway-16346

<?xml version="1.0" encoding="UTF-8"?>

<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="ipe\_ae">

<api>ipe.lightControl&lt;/api>

<rr>true&lt;/rr>

</m2m:ae>

```

\*\*HTTP Response\*\*:

``` http

201 Created

X-M2M-RSC: 2001

X-M2M-RI: home\_gateway-16346

Content-Location: /mn-cse/ae-CAE340304178

```

In response there's Content-Location header which indicates registrated AE address which is necessary to create device resource tree.

AE registration process is comprehensively described in 10.2.2.1 TS-0001 clause.

### 9.3.2 SDT Device resource tree creation in MN-CSE (Resources registration)

This clause describes creation of device resource tree which consists of Device, ModuleClasses and Actions. Firstly, a Device resource is created. Device is mapped to flexContainer resource, so a flexContainer resource is created.

The following request creates flexContainer resource for deviceLight. In response there is a Content-Location header which indicates the new Device address to be used later in ModuleClass creation.

``` http

POST /~/mn-cse/ae-CAE340304178 HTTP/1.1

Host: mncse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml;ty=28

X-M2M-RI: home\_gateway-12

<?xml version="1.0" encoding="UTF-8"?>

<hd:devLt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="deviceLight">

<cnd>org.onem2m.home.device.deviceLight&lt;/cnd>

</hd:devLt>

```

HTTP Response:

``` http

201 Created

X-M2M-RSC: 2001

X-M2M-RI: home\\_gateway-12

Content-Location: /mn-cse/DL3404345178

```

<mark>Reference to TS-0001 is missing</mark>

Flex Container creation procedure is comprehensively described in 10.2.4.16 TS-0001.

To create ModuleClass on Device, the creation request is sent to Device address taken from Device creation request response. In SDT, ModuleClasses are mapped to flexContainer and DataPoints are mapped to customAttribute. To create binarySwitch ModuleClass, the following request is sent.

``` http

POST /~/mn-cse/DL3404345178 HTTP/1.1

Host: mncse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml;ty=28

X-M2M-RI: home\_gateway-43

<?xml version="1.0" encoding="UTF-8"?>

<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="binarySwitch">

<cnd>org.onem2m.home.moduleclass.binaryswitch</cnd>

<powSe type="xs:boolean">false</powSe>

</hd:binSh>

```

HTTP Response:

``` http

201 Created

X-M2M-RSC: 2001

X-M2M-RI: home\_gateway-43

Content-Location: \_/mn-cse/MC43546456

```

In response there's Content-Location header which indicates created ModuleClass address which is used later to create Toggle Action and to modificate state of particular DataPoint.

In SDT, Actions are also mapped to FlexContainer. To add new Action to ModuleClass, the the following request is sent to the ModuleClass address.

``` http

POST /~/mn-cse/MC43546456 HTTP/1.1

Host: mncse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml;ty=28

X-M2M-RI: home\_gateway-44

<?xml version="1.0" encoding="UTF-8"?>

<hd:toggle xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="toggle">

<cnd>org.onem2m.home.moduleclass.binaryswitch.toggle</cnd>

</hd:toggle>

```

HTTP Response:

``` http

201 Created

X-M2M-RSC: 2001

X-M2M-RI: home\_gateway-44

Content-Location: /mn-cse/AC-54783722

```

In response there's Content-Location header which indicates new Action address which is used later to trigger Action (described in clause 9.4.3).

### 9.3.3 Implementation of abstract SDT Device example: deviceLight

This clause presents a "pseudo code" implementation of abstract SDT Device based on deviceLight example. Please note that this pseudo-code is similar to Java but should be considering just an example, it isn't attend to be compilable.

DeviceLight class is an implementation of deviceLight model, it inherits from Device class and has 8 fields relevant for 8 ModuleClasses of deviceLight model. DeviceLight has a contructor which creates objects for ModuleClasses and defines Name and Domain.

```java

/\*\*

\*

\* Stores information about name and domain of the device

\*

\*/

class Device {

string name;

string domain;

public Device(string name, string domain) {

name = name;

domain = domain;

}

}

/\*\*

\* DeviceLight class is an implementation of deviceLight model, it inherits from Device

\* class, and has 8 fields relevant for 8 ModuleClasses of deviceLight model.

\* DeviceLight has a contructor which creates objects for ModuleClasses and defines

\* Name and Domain.

\*/

class DeviceLight extends Device {

FaultDetection faultDetection;

BinarySwitch binarySwitch;

RunMode runMode;

Colour colour;

ColourSaturation colourSaturation;

Brightness brightness;

public DeviceLight(string name, string domain) {

super(name, domain);

}

}

/\*\*

\* Module consists of 3 fields which are relevant to DataPoints parameters in SDT.

\*/

class Module{

boolean optional;

string name;

string domain;

public Module(string name, string domain, boolean optional) {

this.name = name;

this.optional = optional;

this.domain = domain;

}

}

/\*\*

\* BinarySwitch class is considered as an example of ModuleClass implementation,

\* it inherits from Module. It has fields relevant for DataPoints and Actions

\* which are defined in SDT data model. Constructor of BinarySwitch creates powerState

\* DataPoint. BinarySwitch class has also setToggle function which allows to set toogle

\* action which is not mandatory.

\*/

class BinarySwitch extends Module {

BooleanDataPoint powerState;

Toggle toggle;

public BinarySwitch(string moduleName, string moduleDomain, BooleanDataPoint powerStateArgument) {

super(moduleName, moduleDomain, false);

this.powerState = powerStateArgument;

this.powerState.optional = false;

this.powerState.readable = true;

this.powerState.writeable = true;

this.powerState.documentation = "The current status of the binarySwitch. 'True' indicates turned-on, and 'False' indicates turned-off";

}

void setToggle(Toggle toggleArgument) {

this.toggle = toggleArgument;

this.toggle.optional = true;

}

}

/\*\*

\* Colour extends Module and has 3 fields relevant to Colour ModuleClass DataPoints.

\* In the constructor of Colour class there are DataPoints assingments with their

\* parameters. There's also abstract setColour method which allows to set

\* all colours (R,G,B) at once.

\*/

class Colour extends Module {

IntegerDataPoint red;

IntegerDataPoint green;

IntegerDataPoint blue;

public Colour(string moduleName, string moduleDomain, IntegerDataPoint redArgument, IntegerDataPoint greenArgument, IntegerDataPoint blueArgument) {

super(moduleName, moduleDomain, true);

this.red = redArgument;

this.green = greenArgument;

this.blue = blueArgument;

this.red.optional = false;

this.red.readable = true;

this.red.writeable = true;

this.red.documentation = "";

this.green.optional = false;

this.green.readable = true;

this.green.writeable = true;

this.green.documentation = "";

this.blue.optional = false;

this.blue.readable = true;

this.blue.writeable = true;

this.blue.documentation = "";

}

/\*\*

\* This method allows to set all colours (R,G,B) in a single call.

\* Since it is an abstract method it has to be implemented by developer according to

\* specific technology which the device uses.

\*/

public abstract setColour(int R, int G, int B);

}

/\*\*

\* Toggle class consists of 3 fields which are relevant to SDT Action parameters and

\* toggle() method which determines Toggle Action.

\*/

class Toggle {

boolean optional = true;

string documentation;

string name;

public abstract void toggle() { }

public Toggle(boolean optional, string documentation, string name) {

this.optional = optional;

this.documentation = documentation;

this.name = name;

}

}

/\*\*

\* DataPoint class constis of 6 fields relevant to DataPoint SDT parameters.

\* In the constructor there's name and type assingments. The rest of paremeters are

\* assingments in classes which inherits from DataPoint class.

\*/

class DataPoint {

string name;

string type;

boolean optional;

bool readable;

bool writeable;

string documentation;

public DataPoint(string nameArgument, string typeArgument) {

this.name = nameArgument;

this.type = typeArgument;

}

}

/\*\*

\* IntegerDataPoint class consists of two abstract methods getValue() and setValue()

\* which are implemented in HueLight class in 9.3.4 clause.

\* Notice there is DataPoint constructor called in IntegerDataPoint constructor.

\*/

public abstract class IntegerDataPoint extends DataPoint {

public IntegerDataPoint(string name){

super(name, "Integer");

}

abstract int getValue() { };

abstract void setValue(int value) { };

}

/\*\*

\* BooleanDataPoint class consists of two abstract methods getValue() and setValue()

\* which are implemented in HueLight class in 9.3.4 clause.

\* Notice there is DataPoint constructor called in BooleanDataPoint constructor.

\*/

public abstract class BooleanDataPoint extends DataPoint {

public BooleanDataPoint(string name) {

super(name, "Boolean");

}

abstract boolean getValue();

abstract void setValue(boolean value);

}

```

### 9.3.4 Instance of SDT Device example: Hue Light

This section presents a "pseudo code" implementation of IPE for particular device: Hue Light. Please note that this pseudo-code is similar to Java but should be considering just an example, it isn't attend to be compilable.

``` java

/\*\*

\* HueLight class inherits from DeviceLight class. In the contructor of this class

\* DataPoints are created with all needed abstract methods implementation

\* specific for technology (Hue Light in this use case) the device uses. It means that

\* each DataPoint (e.g. powerState in BinarySwitch) has implementation of each abstract \\* method which consists of Hue API calls to control Hue Light.

\*/

class HueLight extends DeviceLight {

public HueLight(string name, string domain) {

super(name, domain);

}

BinarySwitch = new BinarySwitch("binarySwitch", domain,

new BooleanDataPoint("powerState") {

boolean getValue() {

HttpRequest httpRequest;

httpRequest.setURL("http://<bridge\_ip\_address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setMethod("GET");

bool state = parseOnOff(httpRequest.sendHTTPRequest());

return state;

}

void setValue(boolean value){

HTTPRequest httpRequest;

httpRequest.setURL("http://<bridge\_ip\_address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setBody("{\"on\":" + value + "}");

httpRequest.setMethod("PUT");

httpRequest.sendHTTPRequest();

}

});

Colour = new Colour("colour", domain, new IntegerDataPoint("red") {

int getValue() {

HttpRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setMethod("GET");

int red = parseRed(parseColor(httpRequest.sendHTTPRequest()));

return red;

}

void setValue(int value) {

HSVColour hsv = new HSVColour();

hsv.convertFromRGB(value, 0, 0);

HTTPRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" + hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}");

httpRequest.setMethod("PUT")

httpRequest.sendHTTPRequest();

}

}, new IntegerDataPoint("green") {

int getValue() {

HttpRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setMethod("GET");

int green = parseGreen(parseColor(httpRequest.sendHTTPRequest()));

return green;

}

void setValue(int value) {

HSVColour hsv = new HSVColour();

hsv.convertFromRGB(0, value, 0);

HTTPRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" + hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}");

httpRequest.setMethod("PUT")

httpRequest.sendHTTPRequest();

}

}, new IntegerDataPoint("blue") {

int getValue() {

HttpRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setMethod("GET");

int blue = parseBlue(parseColor(httpRequest.sendHTTPRequest()));

return blue;

}

void setValue(int value) {

HSVColour hsv = new HSVColour();

hsv.convertFromRGB(0, 0, value);

HTTPRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" + hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}");

httpRequest.setMethod("PUT")

httpRequest.sendHTTPRequest();

}

}) {

void setColour(int R, int G, int B) {

HSVColour hsv = new HSVColour();

hsv.convertFromRGB(R, G, B);

HTTPRequest httpRequest;

httpRequest.setURL("http://<bridge ip address>/api/1028d66426293e821ecfd9ef1a0731df/lights/1/state");

httpRequest.setBody("{\"on\":true, \"sat\":" + hsv.getSat() + ", \"bri\":" + hsv.getBri() + ",\"hue\":" + hsv.getHue() +"}");

httpRequest.setMethod("PUT")

httpRequest.sendHTTPRequest();

}

};

}

/\*\*

\* SDT Colour model uses RGB colour values and Hue Light device uses HSV model.

\* HSVColour class allows to convert RGB to HSV with its convertFromRGB methods.

\* This class and method isn't a part of SDT data model.

\*/

class HSVColour {

int hue;

int sat;

int value;

void convertFromRGB(int R, int G, int B) {

//Algorithm that converts RGB values to HSV values should be implemented here

}

}

```

## 9.4 Developer of the utility application

### 9.4.0 Introduction

Clause 9.4 describes registration of AE for utility application, discovery process and controlling and monitoring devices.

![Figure 9.4.0-1 Scope of 9.4 Developer of the utility application section](media/ScopeUtilityApp.png)

\*\*Figure 9.4.0-1 Scope of 9.4 Developer of the utility application section\*\*

### 9.4.1 Application Entity registration in IN/MN-CSE

The ADN-AE with IN-CSE registration is shown in the following procedure.

``` http

POST /server?rcn=0 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml;ty=2

X-M2M-RI: server-1234

<?xml version="1.0" encoding="UTF-8"?>

<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone\\_ae">

<api>incse.lightControlApp</api>

<rr>true</rr>

</m2m:ae>

```

HTTP Response:

``` http

201 Created

X-M2M-RSC: 2001

X-M2M-RI: server-1234

Content-Location: /in-cse/ae-CAE345

```

In response Content-Location header there's an address of registered Application Entity.

AE registration process is comprehensively described in 10.2.2.1 TS-0001 clause.

### 9.4.2 Discovery proccess

Clause 9.4.2 describes discovery proccess which uses filter criteria to limit the scope of returned information.

Following response searches for resources with \*cnd\* (Container Definition) set to org.onem2m.home.device.deviceLight so it gives in response list of urils of light devices. Uril is neccessary to control particular device. Fu (filter usage) parameter value set to '1' indicates this is normal resource discovery request so only the addresses of the child resources are returned.

``` http

GET /server?fu=1&cnd=org.onem2m.home.device.deviceLight HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-1234

````

HTTP Response:

<mark>Editor note: XML header is missing</mark>

``` http

200 OK

X-M2M-RSC: 2000

X-M2M-RI: server-1234

Content-Type: application/xml

<m2m:uril xmlns:m2m="http://www.onem2m.org/xml/protocols">

mn-cse/DL3404345178

</m2m:uril>

```

Discovery process is comrehensively described in TS-0001 6.2.5 and 10.2.6 clause.

### 9.4.3 Control & monitor devices

### 9.4.3.0 Introduction

Clause 9.4.3 describes how to control and monitor SDT devices registered in oneM2M network.

### 9.4.3.1 Getting URLs for resources

First there is a need to get a url of ModuleClass resource. ModuleClass is a child of deviceLight, so there is a need to send a request to device address which returns a list of it's children in response. In this particular case there're two: binarySwitch and toggle. Each of them should be controlled by requests sent on given addresses.

Following request gives a list of deviceLight children (ModuleClassess or Actions) in response with their urils.

``` http

GET /~/mn-cse/DL3404345178?rcn=6 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-12345

```

HTTP Response:

<mark>Editor note: response is incorrect</mark>

``` http

200 OK

X-M2M-RSC: 2000

X-M2M-RI: server-12345

Content-Location: /in-cse/ae-CAE345

<?xml version="1.0" encoding="UTF-8"?>

<m2m:fcnt xmlns:m2m="http://www.onem2m.org/xml/protocols">

<ch rn="binarySwitch" ty="28">/mn-cse/MC43546456</ch>

<ch rn="toggle" ty="28">/mn-cse/AC-54783722</ch>

</m2m:fcnt>

```

Here is data from previous request answer which is needed:

```xml

<ch rn="binarySwitch" ty="28">/mn-cse/MC43546456</ch>

<ch rn="toggle" ty="28">/mn-cse/AC-54783722</ch>

```

### 9.4.3.2 DataPoint value retrieving

To monitor powerState, a RETRIEVE request to binarySwitch resource (/mn-cse/MC43546456)is sent as following. In response there is xml payload which contains whole binarySwitch representation.

Remark: Retrieving value of a DataPoint is possible only through retrieving all DataPoints which belong to particular ModuleClass. The set of all ModuleClass's DataPoints is sent in response.

``` http

GET /~/mn-cse/MC43546456?rcn=1 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-12346

```

HTTP Response:

``` http

200 OK

X-M2M-RSC: 2000

X-M2M-RI: server-12346

<?xml version="1.0" encoding="UTF-8"?>

<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols">

<powSe type="xs:boolean">true</powSe>

</hd:binSh>

### 9.4.3.3 DataPoint value changing

To change powerState value, a request to binarySwitch resource (/mn-cse/MC43546456) is sent with xml (or json) representation of new powerState value. PUT request is used because whole resource is updated.

Following requests changes power State Data Point to 'true'.

``` http

PUT /~/mn-cse/MC43546456 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-12346

<?xml version="1.0" encoding="UTF-8"?>

<hd:binSh xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone\_ae">

<powSe type="xs:boolean">true</powSe>

</hd:binSh>

```

HTTP Response:

``` http

204 Updated

X-M2M-RSC: 2004

X-M2M-RI: server-12346

```

### 9.4.3.4 Action triggering

To trigger Toggle action a null content parameter is sent to toggle resource address. (see TS-0023, 6.2.4 Clause, Rule 3-4).

Following request sends null content and triggers Toggle action.

``` http

PUT /~/mn-cse/AC-54783722 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-12346

<?xml version="1.0" encoding="UTF-8"?>

<hd:toggle xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="smartphone\_ae">

</hd:toggle>

```

HTTP Response:

``` http

204 Updated

X-M2M-RSC: 2004

X-M2M-RI: server-12346

```

### 9.4.3.5 Subscription mechanism

Following request creates subscription on binarySwitch resource (mn-cse/MC43546456) which allows to send notifications after its state change. There are two mandatory parameters.

NotificationURI parameter is represented by <nu> xml tag and indicates the address of the device which is supposed to be notificated about ouccured changes. In our case smartphone IP address is used.

Remark: In our use case smartphone IP address is used as a receiver of notifications but in other usage there can be used notification server with static IP address assigned. In such use case this server would be responsible to send notification directly on smartphone address.

NotificationContentType is represented by <nct> xml tag and indicates the type of notification. Its value is one of the following:

- 1: all atributes

- 2: modified attributes,

- 3: resourceID

In our example <nct> is set to 2.

In response there is a whole set of subscription resource parameters.

<mark>Editor note: XML header is missing</mark>

``` http

POST /~/mn-cse/MC43546456 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Content-Type: application/xml

X-M2M-RI: server-12346

<m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="sub\_45346345">

<nu>192.168.10.1</nu>

<nct>2</nct>

</m2m:sub>

```

HTTP Response

<mark>Editor note: wrong response</mark>

``` http

HTTP/1.1 200 OK

X-M2M-RSC: 2000

X-M2M-RI: server-12345

<rn>sub\_45346345</rn>

<ty>23</ty>

<ri>/~/mn-cse/MC43546456/sub-45346345</ri>

<pi>/~/mn-cse/MC43546456/fcnt-45352623</pi>

<ct>20170802T143815</ct>

<lt>20170802T143815</lt>

<nu>192.168.10.1</nu>

<nct>2</nct>

```

After subscription creation, server starts to send verification requests to listener periodically. Verification request has &lt;vrq> attribute set to true. Listener answers with HTTP 200 status to confirm that it is still available. Verification request and listener's response is as follows:

<mark>Editor note: incorrect notification</mark>

``` http

POST 192.168.10.1 / HTTP/1.1

Host: 192.168.10.1

X-M2M-Origin: Soriginator

X-M2M-RI: notif-12345

Content-Type: application/vnd.onem2m-ntfy+xml

<?xml version="1.0" encoding="UTF-8"?>

<m2m:sgn xmlns:m2m="[http://www.onem2m.org/xml/protocols](http://www.onem2m.org/xml/protocols)">

<vrq>true</vrq>

<sud>false</sud>

</m2m:sgn>

```

HTTP Response

``` http

HTTP/1.1 200 OK

X-M2M-RSC: 2000

X-M2M-RI: notif-12345

```

Here is the example of notification which is supposed to be sent after powerState value change. Notification is sent to address which was set in subsription creation request and contains representation of changed DataPoints of binarySwitch resource.

``` http

POST 192.168.10.1 / HTTP/1.1

Host: 192.168.10.1

X-M2M-Origin: Soriginator

X-M2M-RI: notif-12345

Content-Type: application/vnd.onem2m-ntfy+xml

<?xml version="1.0" encoding="UTF-8"?>

<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">

<nev>

<rep>

<powSe type="xs:boolean">true</powSe>

</rep>

<rss>1</rss>

</nev>

<sur>mn-cse/MC43546456/sub-45346345</sur>

</m2m:sgn>

```

HTTP Response

``` http

HTTP/1.1 200 OK

X-M2M-RSC: 2000

X-M2M-RI: notif-12345

```

Following request deletes subscription on binarySwitch.

``` http

DELETE /~/mn-cse/MC43546456/sub-45346345 HTTP/1.1

Host: incse.provider.com:8080

X-M2M-Origin: Soriginator

Accept: application/xml

```

HTTP Response

``` http

HTTP/1.1 204 No Content

```

# 10 Conclusion

A remote light control and monitor use case is proposed to describe IPE and SDT aspects of oneM2M standard. The developer guide described these ascpects from two different perspectives: developer of a device adapter (AE: Inter-working Proxy) and developer of an utility application. The current use case is done by following the high level procedures such as registration of AE with CSE, &lt;flexContainer> resources creation, &lt;flexContainer> retrieveal, &lt;flexContainer> update, subscription and notifications.

# History

|Publication history |Publication history |Publication history |

|-|-|-|

|V1.1.1 |&lt;yyyy-mm-dd> |&lt;CR ID> applied - &lt;Summary of changes> |

|V2.0.0 |12-Mar-2018 |Release 2A - Publication |

| | | |

| | | |

| | | |

| | | |

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