oneM2M
Technical Report

Document Number	TR-0034-V-2.0.0
Document Name:	Developer Guide: CoAP binding and long polling for temperature monitoring
Date:	2018-03-12
Abstract:	The document provides a simple use case for guiding application developers to develop applications using functionalities provided by a oneM2M service platform.

Template Version: 08 September 2015 (Do not modify)

About oneM2M

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

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

Contents

1 Scope

The present document provides a simple use case for guiding application developers to develop applications using functionalities provided by a oneM2M service platform with the scope of as follows:

- objective of the use case;
- the architecture of the use case mapped into an oneM2M service platform;
- the execution procedures for implementation of the use case, and
- implementation details of the use case: CoAP and JSON serialization.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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

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

[i.1] oneM2M Drafting Rules.

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

[i.2] oneM2M TS-0001 (V1.13.0): "Functional Architecture".

[i.3] oneM2M TS-0008 (V1.4.0): "CoAP Protocol Binding".

3 Definitions, symbols and abbreviations

3.1 Definitions

3.2 Symbols

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ADN	Application Dedicated Node
ADN-AE	AE which resides in the Application Dedicated Node
AE	Application Entity
CoAP	Constrained Application Protocol
CSE	Common Services Entity
CSE-ID	Common Service Entity Identifier

DNS	Domain Name System
FQDN	Fully Qualified Domain Name
HTTP	HyperText Transfer Protocol
IN	Infrastructure Node
IN-CSE	CSE which resides in the Infrastructure Node
IP	Internet Protocol
JSON	JavaScript Object Notation
M2M	Machine to Machine
Mca	Reference Point for M2M Communication with AE
Mcc	Reference Point for M2M Communication with CSE
MN	Middle Node
MN-CSE	CSE which resides in the Middle Node
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

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 Use case

This clause briefly describes the use case from perspective of service being provided by the oneM2M platform. The physical device components are introduced in the current clause.

The described use case enables the temperature monitoring function as well as related remote control of actuators via a smartphone or smart tab which embeds an application that gains access to a oneM2M service platform.

An overview of the use case is shown in figure 5-1. The main components include:

- The temperature sensors and actuators are deployed in any place as needed, and connected to a gateway.
- The gateway connected with cloud or a remote server via the necessary communication technologies allowing the temperature sensors and actuators to be managed remotely.
- The cloud or remote server provides a set of services to enable the smartphone to manage the temperature sensors and actuators such as collecting the data of sensors, and switching the actuators on/off, etc.
- The smartphone (or a smart tab) hosts an application to manage the temperature sensors and actuators. The application should support some functions such as discovery the deployed devices, retrieve the data of sensors, and send control commands to actuators. For example, if the data of the temperature sensor indicates the possibility of fire disaster, then the actuator can be triggered to provide water spray or fire alarm.

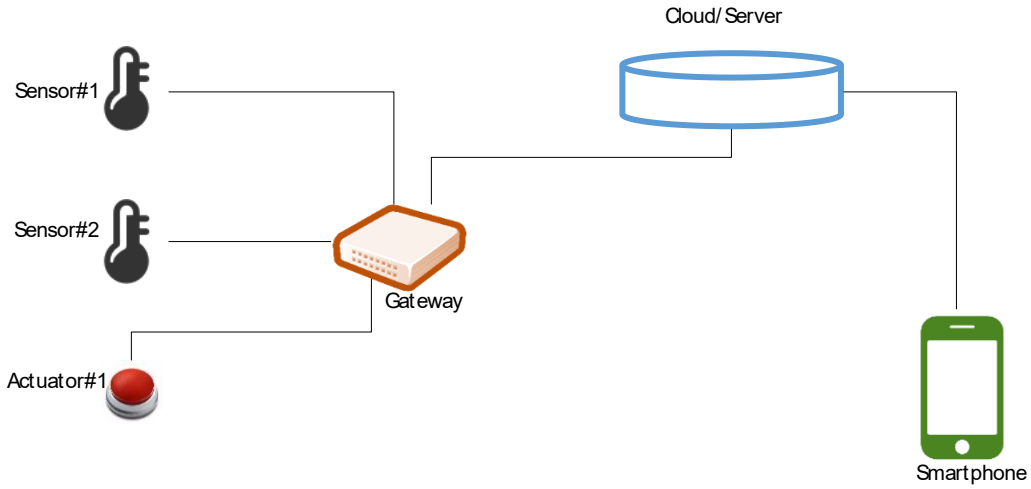

Figure 5-1: Overview of temperature monitoring use case

6 Functional architecture

This clause describes the architecture of this use case with components represented by the oneM2M entity roles.

In the oneM2M, two basic types of entities are defined. One is an AE (short for Application Entity) and the other is a CSE (short for Common Services Entity). Therefore, in this use case:

- The sensor, actuator, and the smartphone each host an AE. The AE which resides in the Application Dedicated Node is called ADN-AE.
- An IN-CSE (short for Infrastructure Node CSE) is hosted in the cloud or server, and a MN-CSE (short for Middle Node CSE) is hosted on the gateway.

For instance, the architecture is show in figure 6-1.

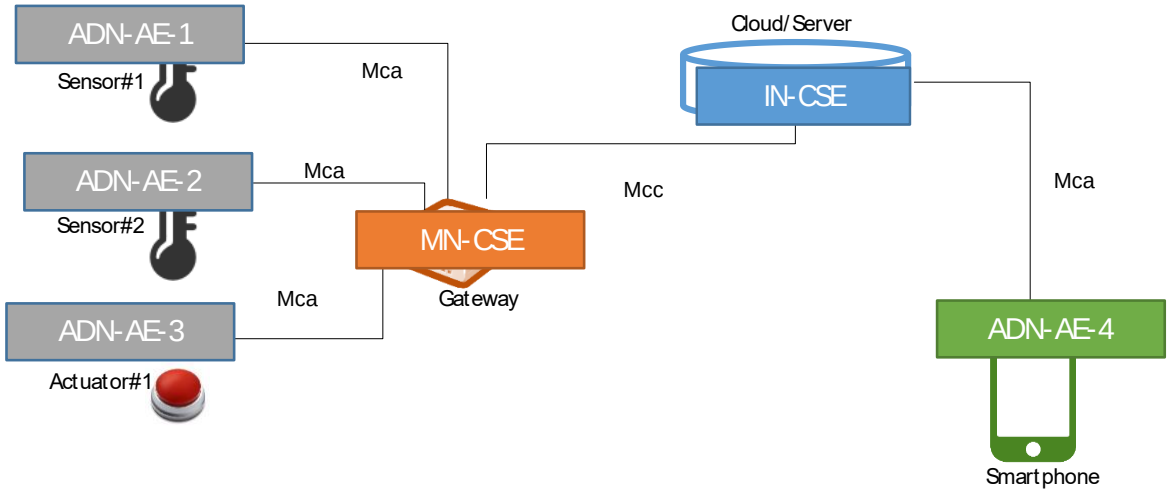

Figure 6-1: oneM2M functional architecture of temperature monitoring use case

The oneM2M defined Mca reference point is used to interface an AE and CSE, and the oneM2M defined Mcc reference point is used to interface CSEs. Therefore, in this use case:

- The reference point used between temperature sensor AE, actuator AE and gateway MN-CSE or Smartphone AE and IN-CSE is Mca.

- The reference point used between the gateway MN-CSE and IN-CSE is Mcc.

In summary, applications used in the current use case are classified as follows:

1) ADN-AE-1: an application embedded in *Sensor#1* with capabilities to monitor *Sensor#1* and interact with the gateway MN-CSE through *Mca* reference point.

2) ADN-AE-2: an application embedded in *Sensor#2* with capabilities to monitor *Sensor#2* and interact with the gateway MN-CSE through *Mca* reference point.

3) ADN-AE-3: an application embedded in *Actuator#1* with capabilities to control *Actuator#1* and interact with the gateway MN-CSE through *Mca* reference point.

4) ADN-AE-4: a smartphone application embedded in the smartphone device with capabilities to interact directly with the oneM2M service platform IN-CSE through *Mca* reference point and thereby remotely monitor and control *Sensor#1, Sensor#2* and *Actuator#1*.

7 Procedures and call flows

7.1 Overviews

In this scenario, user can use the smartphone application to monitor the temperature data of applications embedded in *Sensor#1* and *Sensor#2*. Furthermore, the actuator application embedded in *Actuator#1* also can be managed by the smartphone application. To realize these functions, the deployment of the oneM2M standard in the present use case requires procedures that are classified as follows:

1) **Registration:** the current procedure contains sensor and actuator applications registration, gateway, and smartphone application registration.

2) **Initial resource creation:** the current procedure contains container resource creation.

3) **Discovery and retrieval:** the current procedure contains discovery and retrieval of sensor applications using resource identities through a smartphone application.

In some use cases, if the actuator application embedded in *Actuator#1* is non server capable, which means it cannot be notified by the gateway (MN-CSE). oneM2M defines the <pollingChannel> resource which represents a channel that can be used for such situation [i.2]. Clause 7 also provides relevant procedures and call flows using <pollingChannel> resource:

1) **PollingChannel resource creation:** The current procedure contains pollingChannel resource creation.

2) **Actuator switch on/off:** the actuator application that is discovered by and connected to the smartphone application is able to be switched via polling channel.

7.2 Call flows

7.2.1 Registration

The first step is sensor and actuator application registration, gateway, and smartphone application registration. Typically, sensors and actuators will register applications with the gateway, and then the gateway will register with the oneM2M service platform. The smartphone applications can register with the oneM2M service platform anytime as needed.

Call flows regarding the registration phase depicted in figure 7.2.1-1 are ordered as follows:

1) Sensor applications (ADN-AE1 and ADN-AE2) register with the gateway (MN-CSE).

2) Actuator applications (ADN-AE3) register with the gateway (MN-CSE).

3) Gateway (MN-CSE) registers with the oneM2M service platform (IN-CSE).

4) Smartphone application (ADN-AE4) registers with the oneM2M service platform (IN-CSE).

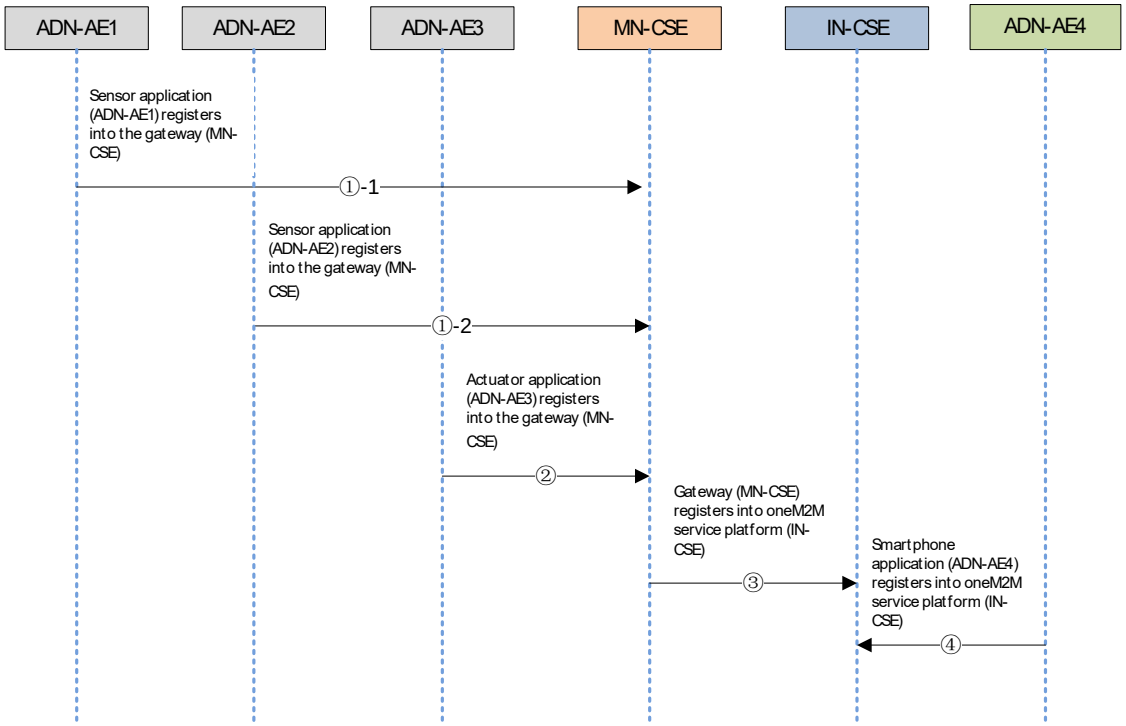

Figure 7.2.1-1: Registration phase call flows

After the initial resource creation process, the resource tree of MN-CSE and IN-CSE is depicted in figure 7.2.1-2.

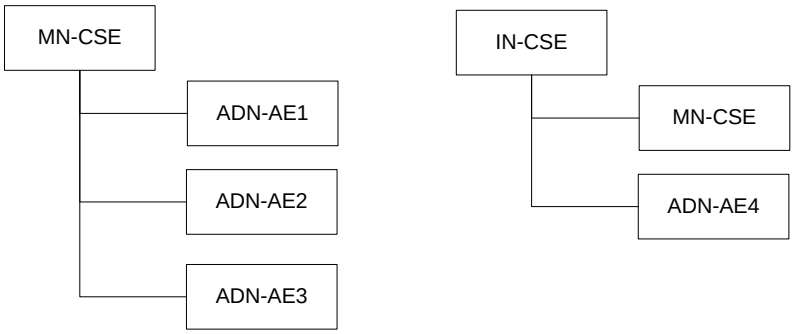

Figure 7.2.1-2: Resource tree of MN-CSE and IN-CSE

7.2.2 Initial resource creation

After registration, it is necessary to create container resources to store the data from sensors on the gateway. Call flows regarding the initial resource creation phase depicted in figure 7.2.2-1 are ordered as follows:

1) Two container resources are created in the gateway (MN-CSE) to store the sensor data under the registered sensor application ADN-AE1 and ADN-AE2, respectively.

2) A container resource is created under ADN-AE3 as a repository of work status of the actuator ADN-AE3.

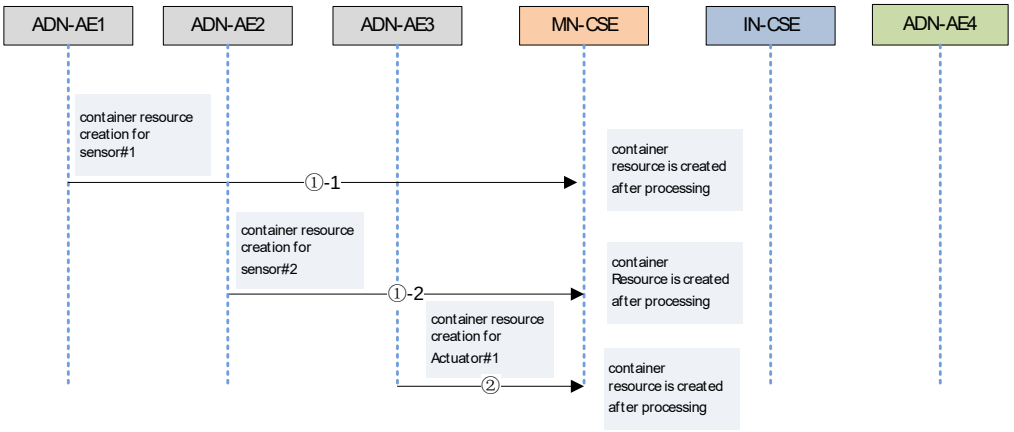

Figure 7.2.2-1: Initial resource creation phase call flows

7.2.3 Discovery and retrieval

Call flows regarding the discovery and retrieval of resources depicted in figure 7.2.3-1 are ordered as follows:

1) The smartphone application (ADN-AE4) periodically sends a RETRIEVE request including the parameter *filterUsage* and specific filter criteria condition(s) as a query string for discovery of resources stored in the MN-CSE of gateway.

2) The gateway (MN-CSE) responds to the smartphone application (ADN-AE4) with URIs of the discovered resources under ADN-AE1, ADN-AE2, if any.

3) The smartphone application (ADN-AE4) sends RETRIEVE requests for retrieval of the latest data from discovered sensor resource, in this example, which is from the container1 of ADN-AE1.

4) The gateway (MN-CSE) responds to the smartphone application (ADN-AE4) with the latest data of *sensor#1*.

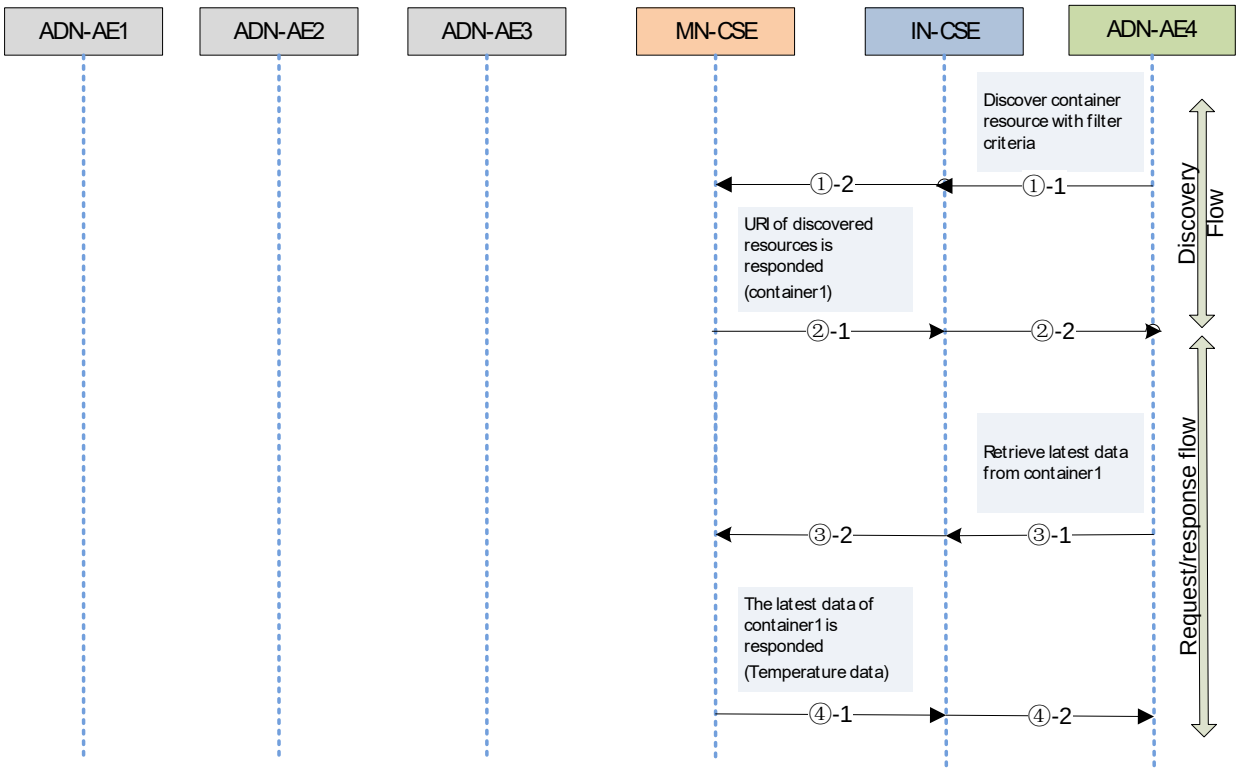

Figure 7.2.3-1: Discovery and retrieval phase call flows

7.2.4 PollingChannel resource creation

As mentioned in the clause 7.1, in some use cases, the actuator application cannot be notified by the gateway (MN-CSE). oneM2M defines the <pollingChannel> resource which represents a channel that can be used for such situation [i.2]. The clause 7.2.4 provides pollingChannel resource creation, as shown in figure 7.2.4-1:

1) A pollingChannel resources is created in the gateway (MN-CSE) to store the actuator status under the registered actuator application ADN-AE3. A pollingChannel resource creation request is initialized by ADN-AE3 target to ADN-AE3 in the MN-CSE. As a result, ADN-AE3 can poll any type of request(s) that targets to itself.

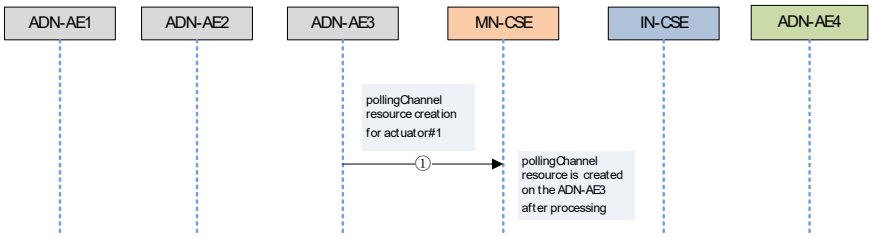

Figure 7.2.4-1: Initial resource creation phase call flows

It is assumed that applications of *sensor#1* and *sensor#2* are registered with MN-CSE via the process in clause 7.2.2. So after the pollingChannel resource creation process, the resource tree of MN-CSE is depicted in figure 7.2.4-2.

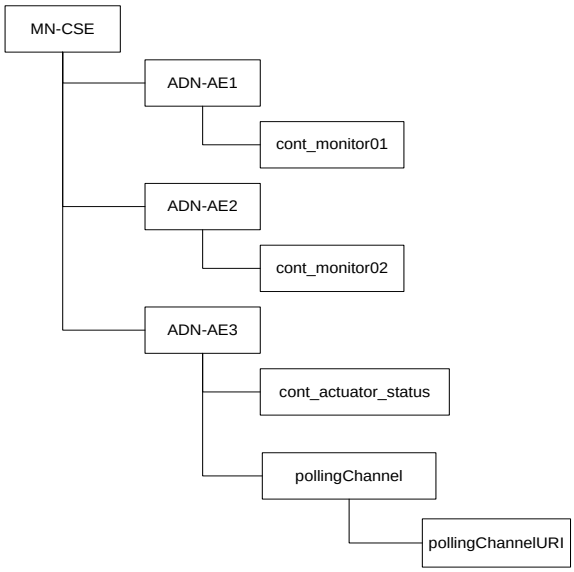

Figure 7.2.4-2: Resource tree of MN-CSE

7.2.5 Actuator switch via polling channel

So far, user can monitor temperature data of *sensor#1* and *sensor#2* by the procedures in clauses 7.2.1, 7.2.2, and 7.2.3. If the data of any temperature sensor is above the threshold, which may indicate the possibility of fire disaster, then the actuator is able to be controlled remotely through the smartphone application accessing the oneM2M service platform and the gateway, in order to achieve some safety management. In the clause 7.2.4, the actuator applications are registered with the gateway (MN-CSE), the smartphone application can discover the actuator application though the same process in the clause 7.2.3. This clause provides the switch process via polling channel.

A call flow for remote control is depicted in Figure 7.2.5-1 and the steps are ordered as follows:

1) The ADN-AE3 sends RETRIEVE request periodically to the gateway in order to retrieve requests, which is marked as REQ1.

2) When the user updates the actuator state on the smartphone, the ADN-AE4 generates a contentInstance create request representing an updated state of actuator ADN-AE3 to the container of ADN-AE3, which is marked as REQ2.

3) The gateway (MN-CSE) internally processes RETRIEVE request and response to the ADN-AE3 with the REQ2 in content parameter.

4) After processing, e.g. turning on the water spray, then the ADN-AE3 sends NOTIFY request REQ3 to the gateway (MN-CSE) carrying the UPDATE response "RESP2" in the content parameter, to indicate the switching requested had been successfully performed.

5) The gateway (MN-CSE) sends UPDATE response to the smartphone application (ADN-AE4), to indicate that the status of actuator is successfully updated.

6) The gateway (MN-CSE) sends NOTIFY response to the actuator application (ADN-AE3), to indicate that the response is successfully sent.

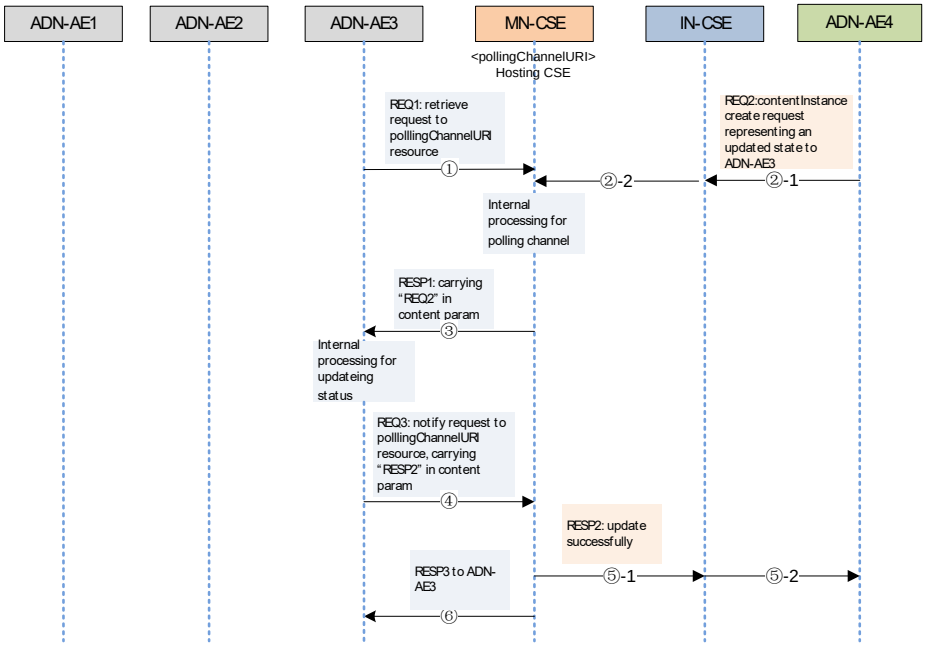

Figure 7.2.5-1: Actuator remote control phase call flows

8 Implementation

8.1 Implementation assumption

Assumptions are presented as below in order to ensure the use case can be correctly implemented.

- Security is not considered in the current use case;
- CoAP binding of oneM2M primitives is used in the current use case, required features according to [i.3];
- JSON serializations of oneM2M primitives is used in the current use case;

- Short names for the representation of the resources and attributes are used in the current use case;

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, using SP-relative structured format.

- IN-CSE:
  - CSE-ID: `/in-cse`
  - Resource ID: `cse127865gu57fa`
  - resourceName of IN-CSE's CSEBase resource: server
- MN-CSE:
  - CSE-ID: `/mn-cse`
  - Resource ID: `cse463432er91er`
  - ResourceName of CSEBase resource: gateway

8.2 Roles of entities

8.2.1 oneM2M service platform (IN-CSE)

The oneM2M service platform is modelled as an IN-CSE and is responsible for:

- handling the requests from smartphone ADN-AE4 and gateway MN-CSE

8.2.2 Sensor applications (ADN-AE1 and ADN-AE2)

Each of the sensor applications are modelled as an ADN-AE and are responsible for:

- initializing the device,
- registering with the MN-CSE,
- creating container resources in the MN-CSE,
- creating content resources under containers sensor1 and sensor2 with data.

8.2.3 Actuator application (ADN-AE3)

The actuator application is modelled as an ADN-AE3 and are responsible for

- initializing the device,
- registering with the MN-CSE,
- creating polling channel resource in the MN-CSE.

8.2.4 Smartphone application (ADN-AE4)

The smartphone application is modelled as ADN-AE4, which directly communicates with the oneM2M service platform IN-CSE and is responsible for:

- initializing the monitor and control application,

- registering the smartphone application with the IN-CSE,
- discovering and displaying,
- accepting and executing the actuator control commands.

8.3 Procedures

8.3.1 Registration and resource creation

The following example shows an sensor application ADN-AE1 registration request and response in clause 7.2.1 using CoAP with JSON serialization.

```
CoAP Request:

Method: 0.02(POST)
Uri-Host: mn.provider.com:5683
Uri-Path: ~
Uri-Path: mn-cse
Uri-Path: gateway
Content-Type: application/vnd.onem2m-res+json
oneM2M-TY: 2
oneM2M-FR: C
oneM2M-RQI: 0001

{
  "m2m:ae":
  {
    "rn": "adn-ae1",
    "api": "001.com.company.temsensor",
    "rr": true
  }
}


CoAP Response:

2.01 Created
oneM2M-RSC: 2001
oneM2M-RQI: 0002
Location-Path: /mn-cse/ae137849axcfrd

{
  "m2m:ae":
  {
    "ty": "2",
    "ri": "ae23456789hgfga",
    "pi": "cse127865gu57fa",
    "ct": "20170327T31415",
    "lt": "20170327T31415",
    "et": "20170927T66666",
    "aei": "CAE23456789"
  }
}
```

Then the following example shows a container create request and response in the procedure of clause 7.2.2 using CoAP with JSON serialization. Result content parameter *rcn* is used and set to 0 to indicate no response is preferred for the CREATE request.

```
CoAP Request:

Method: 0.02(POST)
Uri-Host: mn.provider.com:5683
Uri-Path: ~
Uri-Path: mn-cse
```

```
Uri-Path: gateway
Uri-Path: adn-ae1
Content-Type: application/vnd.onem2m-res+json
oneM2M-TY:3
oneM2M-FR: Cadn-ae1
oneM2M-RQI:0002
Uri-Query: rcn=0

{
  "m2m:cnt":
  {
    "rn": "container1"
  }
}
```

```
CoAP Response:

2.01 Created
oneM2M-RSC: 2001
oneM2M-RQI: 0002
Location-Path: /mn-cse/cont1895qpzlj
```

Then the creation of a content instance resource under the container of ADN-AE1 with initial content is shown in the following procedure. The following example shows a contentInstance create request and response using CoAP with JSON serialization:

```
CoAP Request:

Method: 0.02(POST)
Uri-Host: mn.provider.com:5683
Uri-Path: ~
Uri-Path: mn-cse
Uri-Path: gateway
Uri-Path: adn-ae1
Uri-Path: container1
Content-Type: application/vnd.onem2m-res+json
oneM2M-TY: 4
oneM2M-FR: Cadn-ae1
oneM2M-RQI: 0003

{
  "m2m:cin":
  {
    "cnf": "text/plains:0",
    "con": "23"
  }
}
```

```
CoAP Response:

2.01 Created
oneM2M-RSC: 2001
oneM2M-RQI: 0003
Location-Path: /mn-cse/cin1232gtpaea


{
 "m2m:cin":
 {
    "ty": 4,
    "ri": "cin03243dsfas",
    "pi": "cont1895qpzlj",
    "ct": "20170327T31415",
    "lt": "20170327T31415",
    "et": "20170927T66666",
    "st": 1,
    "cs": 2
```

```
 }
}
```

According to similar procedures, ADN-AE2, ADN-AE3 and necessary resources can register with the gateway as well. Then the gateway can register with the oneM2M service platform. The smartphone applications can register with the oneM2M service platform anytime as needed. Following such resource creation procedures, ADN-AE3 creates a <container> (cont_actuator_status) as its direct child as well as a <subscription> resource is created under the <container> resource (cont_actuator_status).

8.3.2 Discovery and Retrieve

As mentioned in clause 7.2.3, the smartphone application (ADN-AE4) periodically sends a RETRIEVE request including the parameter *filterUsage* and specific filter criteria condition(s) as a query string for discovery of resources stored in the MN-CSE of gateway. It is assumed that the <Container1> resource with label "temp1" is created on MN CSE.

The discovery of containers for each sensors registered with the MN-CSE by the smartphone AE is shown in the following procedure.

```
CoAP Request:

Method: 0.01(GET)
Uri-Host: in.provider.com:5683
Uri-Path: ~
Uri-Path: in-cse
Uri-path: server
Uri-path: gateway
Content-Format: 50(application/json)
oneM2M-FR: Cadn-ae4
oneM2M-RQI: 0004
Uri-Query: fu=1
Uri-Query: ty=3
Uri-Query: lbl="temp1"

CoAP Response:

2.05 OK
oneM2M-RSC: 2000
oneM2M-RQI: 0004


{
 "m2m:uril":
  [
    "gateway/adn-ae1/container1"
  ]
}
```

The smartphone application retrieves URI representing containers registered with MN-CSE from the response message, e.g. gateway/adn-ae1/container1 which is the CSE-Relative structured resource URI format of container.

The smartphone application can retrieve the sensor data from ADN-AE1. If the response is preferred to be returned with a JSON representation, the following is a CoAP request message example:

```
CoAP Request:

Method: 0.01(GET)
Uri-Host: in.provider.com:5683
Uri-Path: ~
Uri-Path: in-cse
Uri-Path: server
Uri-Path: gateway
Uri-Path: adn-ae1
Uri-Path: container1
Content-Format: 50(application/json)
oneM2M-FR: Cadn-ae4
```

```
oneM2M-RQI:0005

CoAP Response:

2.05 OK
oneM2M-RSC: 2000
oneM2M-RQI: 0005
Content-format: application/vnd.onem2m-res+json

{
 "m2m:cin":
 {
    "ty": 4,
    "ri": "cin0276fd56fd",
    "pi": "cont1895qpzlj",
    "ct": "20170327T31415",
    "lt": "20170327T31415",
    "et": "20170927T66666",
    "st": 1,
    "cnf": "text/plain:0",
    "cs": 2,
    "con": "23"
 }
}
```

8.3.3 Long Polling

As mentioned in clause 7.2.4, for using long polling procedures, ADN-AE3 creates <pollingChannel> resource under its <AE> resource on MN-CSE. <pollingChannelURI> is a virtual resource and made automatically by Hosting CSE during creating parent <pollingChannel>. The following example shows <pollingChannel> resource create request and response using HTTP with JSON serialization.

```
HTTP Request:

POST /~/mn-cse/gateway/adn-ae3?rcn=0 HTTP/1.1
Host: mn.provider.com:8080
X-M2M-Origin: Cadn-ae3
Content-Type: application/vnd.onem2m-res+json;ty=15
X-M2M-RI: mncse-12345

{
 "m2m:pch":
 {
    "rn":"pch_actuator01"
 }
}


HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: mncse-12345
Content-Location: /mn-cse/pch7890afer34
```

ADN-AE3 also generates a container and a content instance resource under the ADN-AE3 to store the work status of the actuator ADN-AE3, following to the similar procedure in clause 8.3.1. It is assumed that a subscription to this container resource has been created where a notificationURI attribute is set to the resource identifier of <pollingChannel>.

Then user can subscribe or make a change of the actuator state on the gateway. When users make a change to the actuator state via the smartphone user interface, the smartphone application (ADN-AE4) performs a new content Instance creation procedure carrying the new state in request and targeting to the ADN-AE3.

If the contentInstance create request body is represented in JSON, the following is a HTTP request message example:

```
HTTP Request:
```

```
POST /~/mn-cse/gateway/adn-ae3/cont_actuator_status?rcn=0 HTTP/1.1
Host: mn.provider.com:8080
X-M2M-Origin: Cadn-ae4
Content-Type: application/vnd.onem2m-res+json;ty=4
X-M2M-RI: mncse-11123

{
  "m2m:cin":
  {
    "cnf": "text/plains:0",
    "con": "ON"
  }
}

HTTP Response:

201 Created
X-M2M-RSC: 2001
X-M2M-RI: mncse-11123
Content-Location: /mn-cse/cin7893setj34
```

Then the gateway will generate a notification based on the notification event criteria that is a new content instance creation under the subscribed container resource (`cont_actuator_status`). Usually the actuator (ADN-AE3) sends periodically a Retrieve request to the <pollingChannelURI> resource on the gateway in order to retrieve the notification target to itself.

If the request body is represented in JSON, the following is a HTTP request message example:

```
HTTP Request:

GET /~/mn-cse/gateway/adn-ae3/pch_actuator01/pcu HTTP/1.1
Host: mn.provider.com:8080
X-M2M-Origin: Cadn-ae3
X-M2M-RI: mncse-11223


HTTP Response:

200 OK
X-M2M-RSC : 2000

X-M2M-RI: mncse-11223

{
 "m2m:sgn":
 {
    "nev":{
       "rep":
       {
          "cin":
          {
             "cnf": "text/plain:0",
             "con": "ON"
          }

       },
       "net": [3]
    },
    "sur":"/mn-cse/sub856463ahyvc28"
 }
}
```

After retrieved the notification, the states of actuator (ADN-AE3) can be updated automatically. Then the ADN-AE3 sends request to the gateway to indicate the previous notification had been successfully performed.

9 Conclusions

The current use case is realized by following the high level procedures such as registration of smart devices, gateway with the oneM2M service platform, container resource creation, content instance retrieval, and using polling channel for actuator switch.

XML serialization and HTTP binding of oneM2M primitive, as well as other implementation examples are intended to be covered in other developer guidances.

X

History

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