International Telecommunication Union

ITU-T Technical Report

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(15 September 2017)

YSTR-M2M-DG.DM
oneM2M – Developer guide of device management

Technical Report ITU-T YSTR-M2M-DG.DM

oneM2M – Developer guide of device management

Summary
This Technical Report provides a developer guide for device management using oneM2M and includes:

– device management using oneM2M protocol;

– device management using oneM2M and other management technologies.

History

This document contains Version 0 of the ITU-T Technical Report on "oneM2M-Developer guide of device management" approved at the ITU-T Study Group 20 meeting held in Geneva, 4-15 September 2017.

Keywords

Developer guide, device management, oneM2M.

Table of Contents

Technical Report ITU-T YSTR-M2M-DG.DM

oneM2M – Developer guide of device management

1 Scope

This Technical Report provides examples of message flow and procedures on how device management is performed using oneM2M defined primitives and procedures.

2 References

None.

3 Terms and definitions

3.1 Terms defined elsewhere

None.

3.2 Terms defined in this Technical Report

None.

4 Abbreviations and acronyms

This Technical Report uses the following abbreviations and acronyms:

ADN Application Dedicated Node

AE Application Entity

ASN Application Service Node

CSE Common Services Entity

IN Infrastructure Node

IN-CSE Infrastructure Node-CSE

LWM2M Lightweight M2M

MN Middle Node

MN-CSE Middle Node-CSE

NoDN Non-oneM2M Device Node

5 Conventions

None.

6 Device management over the service layer

6.1 Introduction

For device management, oneM2M uses the <mgmtObj> resource. For device management over the service layer, the <mgmtObj> that is used for the management of the managed entity is located at different places depending on the characteristic of the managed entity. In any case, there exists one driver or client on the managed entity that observes the update of the <mgmtObj> resource to perform management operation.

For application service node (ASN), middle node (MN) and infrastructure node (IN) which has common services entity (CSE), the <mgmtObj> is hosted on the corresponding ASN-CSE, MN-CSE and IN-CSE as child resources of the <node> resource that represents the ASN, MN and IN. The managed entity observes the <mgmtObj> directly to perform the management operation.

For application dedicated node (ADN), the <mgmtObj> resource is hosted on the Registrar CSE of the application dedicated node-application entity (ADN-AE) as child resource of the <node> resource that represents the ADN. The ADN-AE in this case should subscribe the <mgmtObj> resources under the corresponding <node> resource to receive notifications on any update of the <mgmtObj> resource. The ADN which is the managed entity in this case further performs the management operation based on the notifications received.

For non-oneM2M device node (NoDN), the <mgmtObj> resource is hosted on the CSE which the NoDN is connected to. As NoDN is not a oneM2M defined entity, the connection between NoDN and the CSE is out of scope of oneM2M and is implementation specific. In this case, the <mgmtObj> resource on the CSE under the corresponding <node> resource should be monitored to determine if any management operation needs to be performed. This part is implementation specific and is out of scope of oneM2M as well.

6.2 Use case

An application wants to retrieve the current available memory of the device. In this case, the specialization of <mgmtObj> memory is used, see clause D.4 of [b-ITU-T Y.4500.1]. In the memory specialization, *memAvailable* is the attribute that contains the required information. The application can get the available memory by retrieving the resource. See Figure 6.2-1.

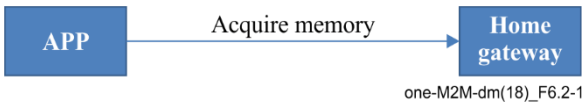

Figure 6.2-1 – Use case for management

6.3 Architecture

6.3.1 Management of ASN, MN and IN

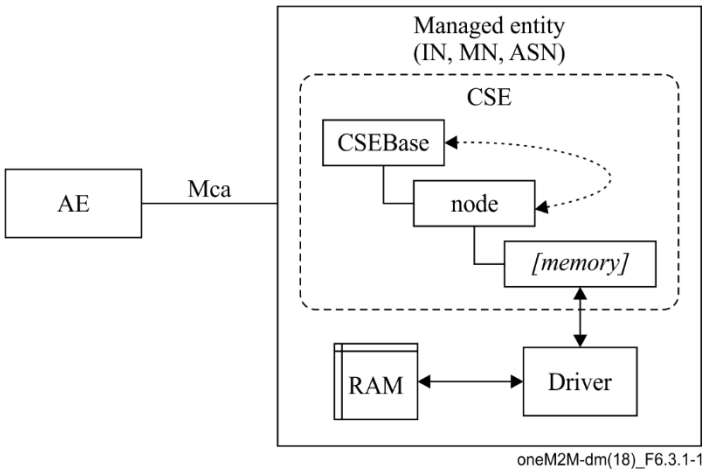

Figure 6.3.1-1 – Architecture for management of IN,MN,ASN

Figure 6.3.1-1 shows the architecture for management of IN, MN and ASN. As these managed entities have their own CSEBase resource, they have the capability to host oneM2M resources. When using device management over the service layer, the *[memory]* resource is directly hosted under the <node> resource of the CSEBase. In this case, the <node> resource represents the IN, MN, ASN themselves.

The driver is the software that interacts with the memory of the device that acquires the total memory and available memory from the system.

In this case, the *[memory]* specialization is modified by the driver using an internal interface. The modification may be done periodically or with some other policies which are out of scope of the oneM2M standard.

Whenever the AE issues retrieve to the *[memory]* specialization, the value from the resource is returned.

6.3.2 Management of ADN

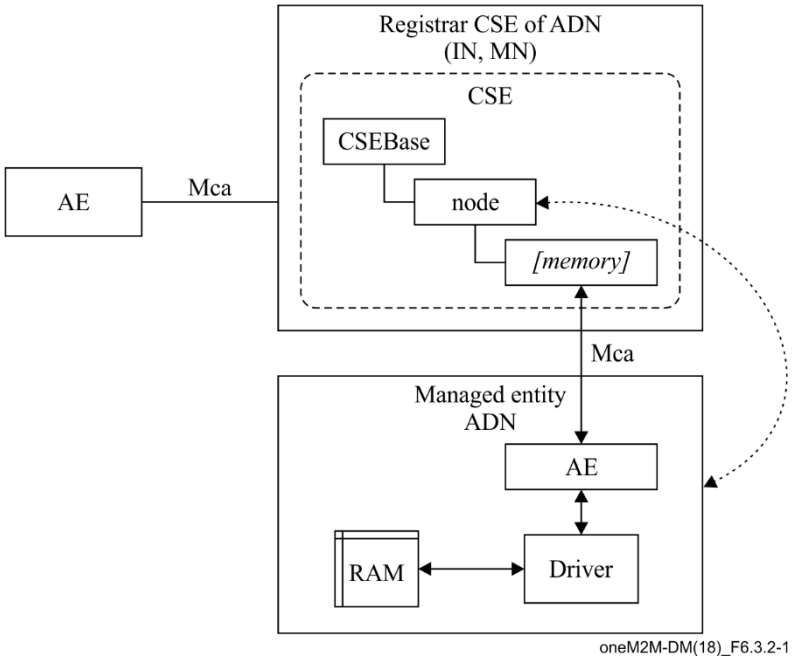

Figure 6.3.2-1 – Architecture for management of ADN

Figure 6.3.2-1 shows the architecture of management of and where the *[memory]* specialization is hosted on the Registrar CSE of the ADN. According to the supported configuration of oneM2M architecture, the Registrar CSE could be IN-CSE or MN-CSE.

In this case, the <node> resource hosted under the CSEBase of the Registrar CSE represents the ADN which is the managed entity. The driver retrieves memory information using internal interfaces and makes the ADN-AE update the memory value to the *[memory]* specialization over the Mca reference point. The update over Mca is triggered periodically or based on the policy of the ADN which is implementation specific.

6.3.3 Management of NoDN

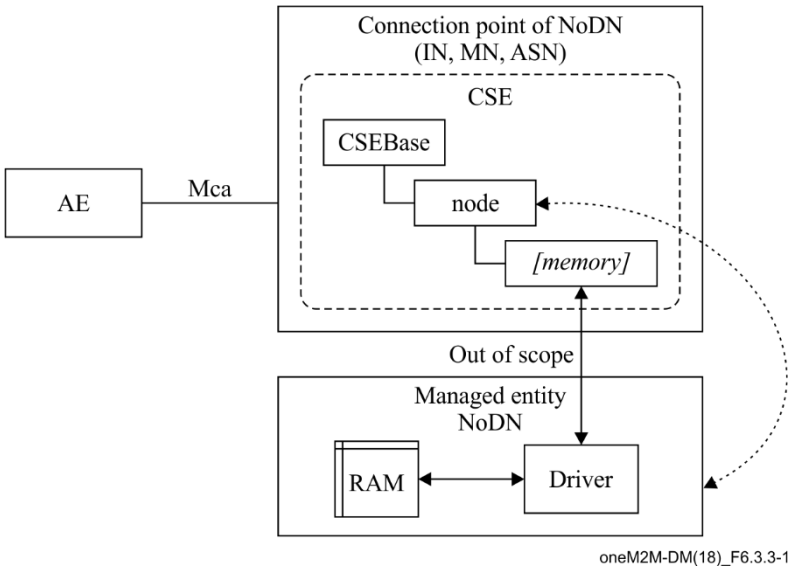

Figure 6.3.3-1 – Architecture for management of NoDN

Figure 6.3.3-1 shows the architecture of management of the NoDN where the *[memory]* specialization is hosted on the connection point of the NoDN. The connection point is the entity that the NoDN is connected to. The entity has an adaptor that shares the same network protocol with the NoDN.

In this case, the <node> resource hosted under the CSEBase represents the NoDN. The connection between the driver and the CSE is out of scope of oneM2M. It may be Bluetooth, ZigBee or even proprietary.

The update of the memory information happens over the interface between the driver and the CSE which is out of scope of oneM2M.

6.4 Procedures

6.4.1 Management of ASN, MN and IN

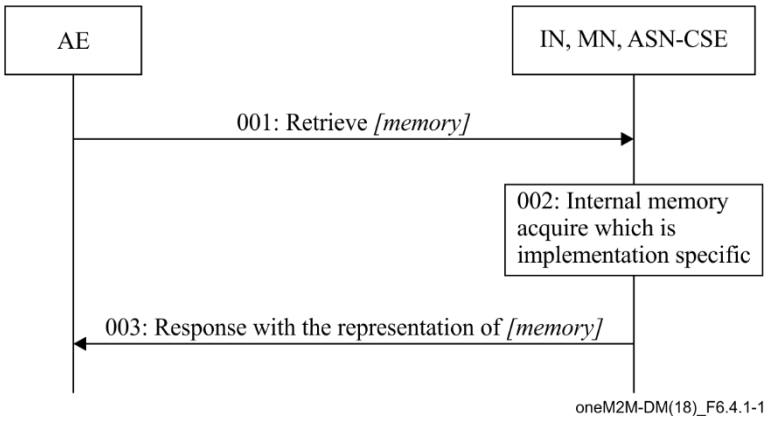

Figure 6.4.1-1 – Procedures for management of IN, MN, ASN

Figure 6.4.1-1 shows the procedures for management of IN, MN, and ASN as follows:

001: The AE send an oneM2M Retrieve primitive to the ResourceID of *[memory]* resource.

002: The CSE receives the Retrieve primitive and triggers the driver to acquire the current available memory value and update the *[memory]* resource.

003: The CSE responds to the AE with the representation of the *[memory]* resource.

NOTE – The AE could also subscribe the [memory] resource to get the notification on the update of the [memory] resource.

The [memory] resource in this case is locally created beforehand by the Hosting CSE using internal interfaces.

6.4.2 Management of ADN

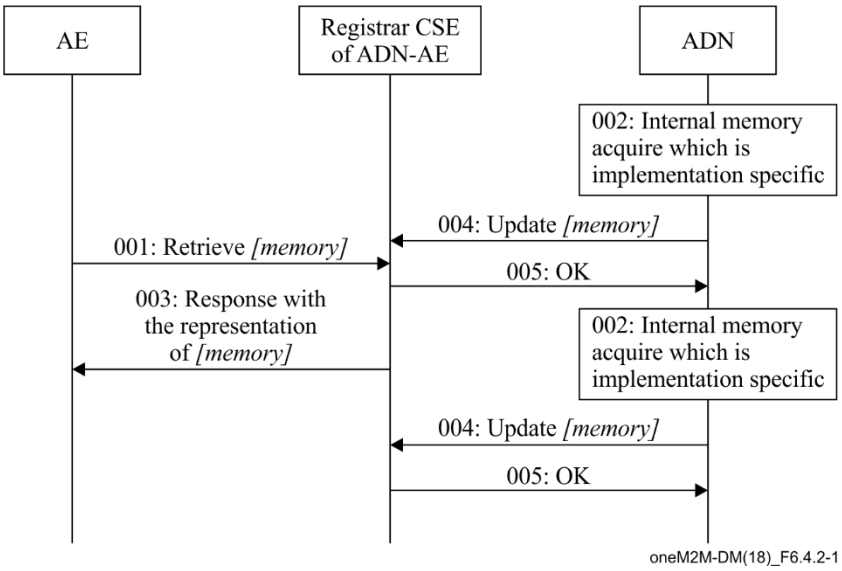

Figure 6.4.2-1 – Procedures for management of ADN

Figure 6.4.2-1 shows the procedures for management of AND as follows:

001: The AE sends a Retrieve request to Registrar CSE of ADN-AE. Requesting the *[memory]* resource.

003: The Registrar CSE responds with the representation of the *[memory]* resource.

002: The ADN internally acquires the available memory information.

004: The ADN-AE updates the *[memory]* resource using the Mca reference point.

005: Update successful.

NOTE – 001, 003 and 002, 004 do not have a specific order. The ADN is updating the *[memory]* resource periodically. In 003, the Registrar CSE always responds with the current resource.

Subscription and notification could also be used by the AE to monitor the current status of the *[memory]* resource.

The [memory] resource in this case is created by the ADN-AE.

6.4.3 Management of NoDN

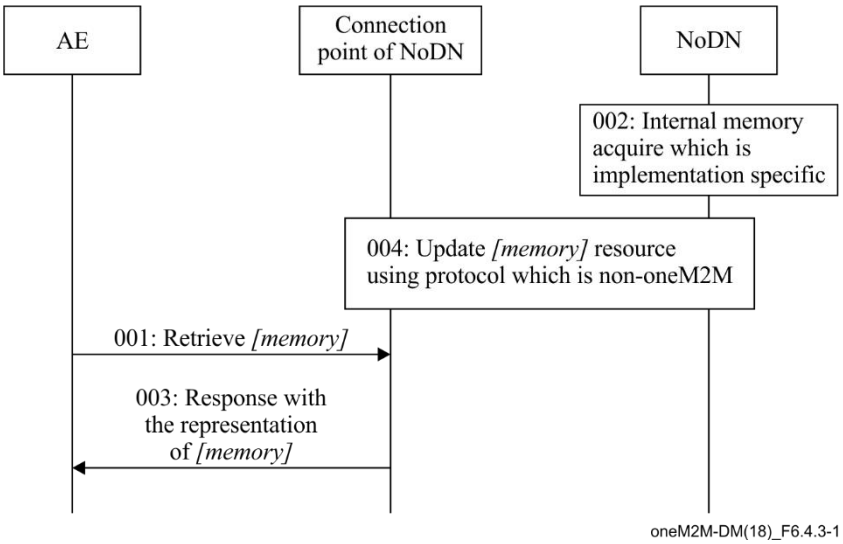

Figure 6.4.3-1 – Procedures for management of NoDN

Figure 6.4.3-1 shows the procedures for management of NoDN as follows:

001: The AE sends Retrieve request to the connection point of the NoDN. Requesting the *[memory]* resource.

003: The CSE of the connection point responds with the representation of the *[memory]* resource.

002: The NoDN internally acquires the available memory information.

004: The NoDN updates the *[memory]* resource using a protocol that is non-oneM2M.

NOTE – 001, 003 and 002, 004 do not have a specific order. The NoDN is updating the *[memory]* resource periodically. In 003, the Registrar CSE is responding always the current resource.

The [memory] resource in this case is created by the NoDN via a non-oneM2M interface.

6.5 Implementation

6.5.1 Introduction

In this implementation, the current Technical Report uses HTTP binding and JSON serialization as examples, and service provider (SP)-relative-structured resource addressing format is used.

The following resource ID and/or entity ID is used:

– Memory resource
  - Resource Name `"memory001"`
  - Resource ID "m343245334adf" represented in CSE-relative Unstructured-Resource-ID of the [memory] resource
  - Parent Resource ID `"n893051036jdg"`
– AE-ID "C324352841DAS" of the managed entity
– Registrar/Hosting CSE
  - SP-relative CSE-ID "/CSE0034234"
  - CSE Resource Name "server"
– Host name of the Registrar/Hosting CSE "in.management.server.com"

6.5.2 Management of ASN, MN and IN

Step	Message Example

001	HTTP Request
	`GET /~/CSE0034234/server/memory001 HTTP/1.1`
	`Host : in.management.server.com`
	`X-M2M-RI : 1234`
	`X-M2M-Origin: C324352841DAS`
	`Accept : application/json`

002	HTTP Response
	`200 OK`
	`X-M2M-RI : 1234`
	`X-M2M-RSC : 2000`
	
	`{`
	`"m2m:mgo" : {`
	`"ri":"m343245334adf",`
	`"pi":"n893051036jdg",`
	`"ct":"20161129T121550",`
	`"et":"20181231T235959",`
	`"lt":"20161130T152341",`
	`"ty":13,`
	`"rn":"memory001",`
	`"mgd":1003,`
	`"mma":800,`
	`"mmt":1024`
	`}`
	`}`

6.5.3 Management of ADN

Step	Message Example
001	Same as 001 in 6.5.2
002	Same as 002 in 6.5.2

004	HTTP Request
	`PUT /~/CSE0034234/server/memory001?rcn=0 HTTP/1.1`
	`Host : in.management.server.com`
	`X-M2M-RI : 1235`
	`Content-Type: application/json`
	`X-M2M-Origin: C324352841DAS`
	
	`{`
	`"m2m:mgo":{`
	`"mma":600`
	`}`
	`}`

005	HTTP Response
	`200 OK`
	`X-M2M-RI : 1235`
	`X-M2M-RSC : 2004`

6.5.4 Management of NoDN

001 and 003 are the same as in clause 6.5.2.

002 and 004 are protocol specific and even proprietary, thus are out of scope of oneM2M.

7 Device management using external management technologies

7.1 Introduction

In the case of device management using external management technologies such as open mobile alliance (OMA) device management (DM), lightweight M2M (LWM2M) or broadband forum (BBF) TR069. All management is done through the IN-CSE, since only the IN-CSE has the capability to talk to the management server. Therefore, in this case, all <node> resources that represent the managed entity are hosted under the <CSEBase> of the IN-CSE. The manged entity may be IN, MN, ASN, ADN or NoDN. Then the <mgmtObj> that is used for management is hosted as the child resource of <node> resource. The attribute *objectIDs* and *objectPaths* of the <mgmtObj> are set in this case to map to the external management objects.

7.2 Use case

An application wants to retrieve the current avaiable memory of the device. In this use case, the specialization of <mgmtObj> memory is used as in clause D.4 of [b-ITU-T Y.4500.1]. In the memory specialization, *memAvailable* is the attribute that contains the required information. The application can get the available memory by retrieving the resource. See Figure 7.2-1.

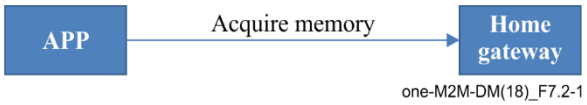

Figure 7.2-1 – Use case for management

7.3 Architecture

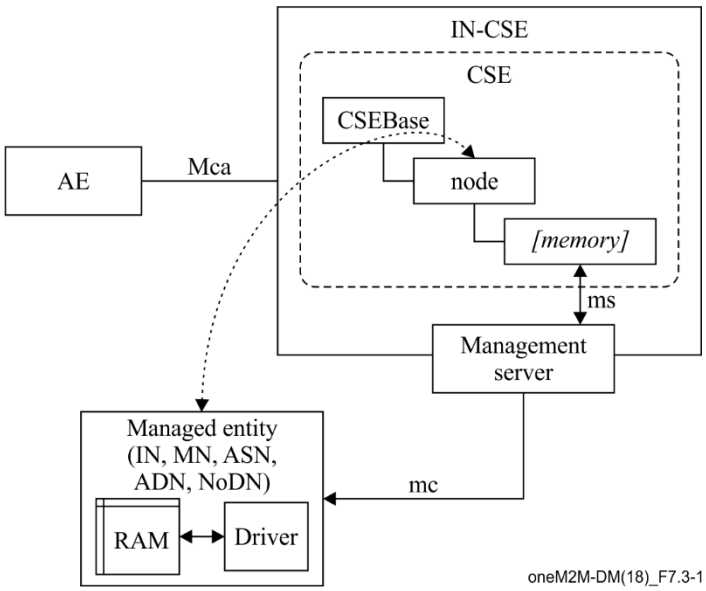

Figure 7.3-1 – Architecture for management using external technology

Figure 7.3-1 shows the case of architecture for management using external management technology where all management related resources are hosted on the IN-CSE. The IN-CSE interacts with the management server through ms interface and triggers the management server to send management commands to the managed entity through the mc interface. The mc and ms interfaces are defined by external management technology or are implementation specific which is out of scope of oneM2M.

The management server may be deployed together with the IN-CSE, in this case, the IN-CSE directly sends management commands through the mc interface to the managed entity. The management server may also be deployed separately from the IN-CSE; in this case, the IN-CSE need to send

commands to the management server to trigger the management server to send management commands to the managed entity. The ms interface may be implementation specific.

7.4 Procedures

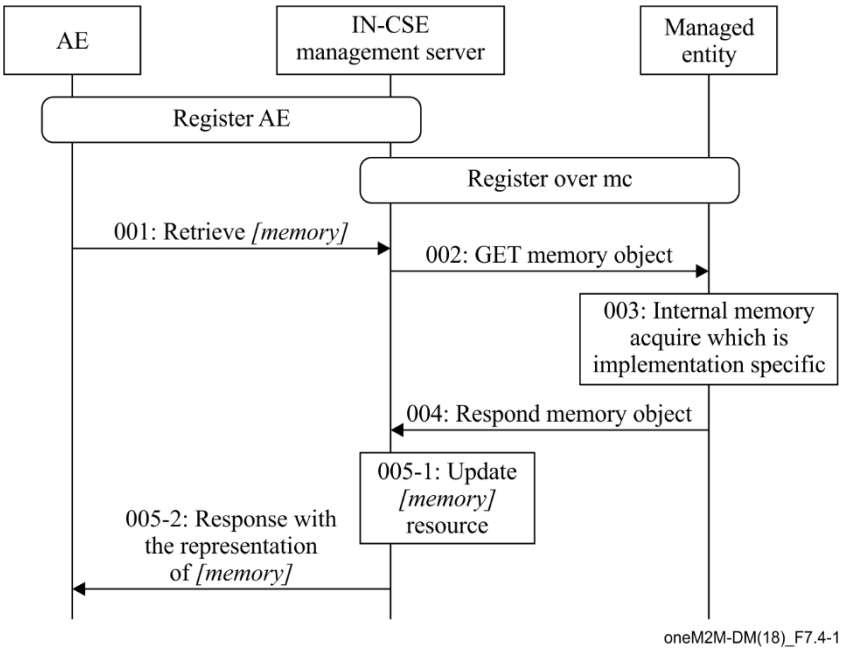

Figure 7.4-1 – Procedures for management to ADN

Figure 7.4-1 shows the procedures for management to AND as follows:

001: The AE retrieves *[memory]* resource specialization from the IN-CSE over the mca interface.

002: On receiving the request, the IN-CSE sends management command to managed entity using external management technology implementation.

003: On the managed entity, the driver acquires the memory information.

004: The managed entity responds with acquired memory information over the mc interface.

005: The IN-CSE updates the *[memory]* resource and respond the RETRIEVE request with the representation of *[memory]* resource.

7.5 Implementation

In the implementation, LWM2M is used as the external management technology. Resource ID and/or entity ID that are defined in clause 6.5.1 is also applicable to this clause.

In addition, the host name of the managed entity is named "adn.managed.entity.com", the memory resource name is "memory002", the memory resource ID is "m463245245afg", and the parent resource ID of the memory resource is "n993051028asd". The request URL for the managed entity is assumed as "coap://adn.managed.entity.com/adn001/memory".

Step	Message example

001	HTTP Request
	`GET /~/CSE0034234/server/memory002 HTTP/1.1`
	`Host : in.management.server.com`
	`X-M2M-RI : 3454`
	`X-M2M-Origin: C324352841DAS`
	`Accept : application/json`

Step	Message example

002	CoAP Request
	`Method : 0.01 (GET)`
	`Uri-Host : adn.managed.entity.com`
	`Uri-Path : adn001`
	`Uri-Path : memory`
	`oneM2M-FR : C324352841DAS`
	`oneM2M-RQI: 4553`
	`Payload : empty`

004	CoAP Response
	`X-M2M-RSC : 2.05 (OK)`
	`oneM2M-RQI : 4553`
	`Content-Format: 50 (application/json)`
	`Payload : {memory_object_serialized_in_JSON}`

005-2	HTTP Response
	`X-M2M-RI : 3454`
	`X-M2M-RSC : 2000`
	`{`
	`"m2m:mgo" : {`
	`"ri":"m463245245afg",`
	`"pi":"n993051028asd",`
	`"ct":"20161129T121550",`
	`"et":"20181231T235959",`
	`"lt":"20161130T152341",`
	`"ty":13,`
	`"rn":"memory002",`
	`"mgd":1003,`
	`"mma":800,`
	`"mmt":1024`
	`}`
	`}`

Bibliography

[b-ITU-T Y.4500.1] Recommendation ITU-T Y.4500.1 (2018), *oneM2M – Functional architecture.*