ETSI TR 103 841V0.0.1 (2024-01)

SmartM2M; oneM2M Performances Evaluation Tool

(Proof of Concept)

**Technical Report**

Reference

DTR/SMARTM2M-103841

Keywords

oneM2M, performance evaluation, open source, simulation, KPI

***ETSI***

650 Route des Lucioles

F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C

Association à but non lucratif enregistrée à la

Sous-Préfecture de Grasse (06) N° 7803/88

***Important notice***

The present document can be downloaded from:  
<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at <https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:  
<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

***Copyright Notification***

Reproduction is only permitted for the purpose of standardization work undertaken within ETSI.  
The copyright and the foregoing restrictions extend to reproduction in all media.

© ETSI 2020.

All rights reserved.

**DECT™**, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.  
**3GPP™**and **LTE™** are trademarks of ETSI registered for the benefit of its Members and  
of the 3GPP Organizational Partners.  
**oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and  
of the oneM2M Partners.  
**GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Contents

Intellectual Property Rights 4

Foreword 4

Modal verbs terminology 4

Introduction 4

1 Scope 5

1.1 Context for the present document 5

1.2 Scope of the present document. 5

2 References 6

2.1 Normative references 6

2.2 Informative references 6

3 Definition of terms, symbols and abbreviations 7

3.1 Terms 7

3.2 Symbols 7

3.3 Abbreviations 7

4 oneM2M profiler 8

4.1 Structure 8

4.2 Operating system interaction 8

4.3 Installation and configuration 8

5 oneM2M simulator 8

5.1 OMNeT++ at the glance 8

5.2 oneM2M deployment model 8

5.3 Mapping on ONMeT++ library and the ned topology 8

5.4 Installation and configuration 8

6 Performance evaluation framework at work 8

6.1 Mapping Scenario Description 8

6.2 Synthetising KPIs by the profiler run 9

6.3 Injecting KPIs in the OMNeT++ simulator 9

6.4 Performance evaluation analysis 9

7 Conclusions, improvement, extensions 9

Annex A: Source code 9

Annex B: Change history 9

# Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<https://ipr.etsi.org/>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

# Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

# Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](https://portal.etsi.org/Services/editHelp!/Howtostart/ETSIDraftingRules.aspx) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# Introduction

TTF 019 aims are studying, analysing, evaluating, and simulating IoT application deployments on some oneM2M open-source implementations. Evaluation will be conducted on the basis of case studies, deployment model, and performance KPIs, all described in previous ETSI TR 103 839 [i.1] and ETSI TR 103 840 [i.2].

The object of the present document [i.3] is to describe and present the simulation tool and the profiler tool developed to that aim. More precisely, the simulation tool is a OMNeT++ library implementing either the deployment model and the case studies as described in [i.1,i.2]; the profiler is a standalone software to be runned together with a real open-source oneM2M implementation, providing execution KPIs values: these values will be injected in the OMNeT++ simulator allowing a realistic large scale simulations.

# 1 Scope

## 1.1 Context for the present document

The oneM2M ETSI standard is now mature: multiple deployments exist all over the world at both experimental and operational levels. The experimental deployments are conducted for multiple reasons:

* to evaluate the capabilities of the standard in terms of expressiveness, usability on specific equipment, connection with specific existing systems or performance evaluation.
* to provide a methodological study, based on performance evaluation (time, space) on a given set of "paradigmatic use cases".
* to measure KPIs defined in this document*.* Different implementations exist that are compliant with the oneM2M standard, available either freely or commercially.

Use cases are evaluated in terms of chosen KPI: e.g., running time, memory space, numerosity of oneM2M entities (e.g., AE, MN-CSE, CSE.), data transfer volume and real-time needs. Using a select set of available oneM2M CSE implementations, a simulation library or an *ad hoc* simulator is to be provided, offering the ability to evaluate and simulate the performance of the use cases and give crucial information/feedback to the general user of the oneM2M to choose and tune their IoT applications based on oneM2M framework. The results of this tool development and evaluations of the use cases will be the basis to generate other deliverables. The present document was developed in the context of ETSI TTF T019, set up to perform work on "Performance Evaluation and Analysis for oneM2M Planning and Deployment". Five elements were addressed sequentially:

1. a collection of **use cases and derived requirements** were formally identified and defined. This work includes identification of relevant deployment scenarios. We adopted the use case style and template from oneM2M with a minor modification to address some performances issues. This phase of the work resulted in deliverable ETSI TR 103 839 [i.1].
2. The definition of **performance evaluation model**, with specification of procedures to assess the performance of oneM2M-based IoT platforms. This includes the identification and definition of a set/list of KPIs necessary to assess the deployment. For those KPIs, provision of a formal description of the test campaign and the test results to be obtained. This phase of the work resulted in deliverable ETSI TS 103 840 [i.2] (the present document).
3. The creation of a **proof of concept** of a performance evaluation tool. This work also relies on a formal description of the identified deployment scenarios (single vertical domain & multiple vertical domains). This phase of the work resulted in deliverable ETSI TR 103 841 [i.3].
4. A practical **demonstration and analysis** exercise putting the proposed tool to use, with a specific oneM2M implementation but aimed at being a blueprint for the adoption and re-use of the results of TR 103 839, TS 103 840, and TR 103 841 with other oneM2M implementations and deployment scenarios. This phase of the work resulted in deliverable ETSI TR 103 842 [i.4].
5. The development of a set of **guidelines and best practices** documenting best practices and lessons learnt as well as providing instructions for IoT solution topology, capacity provisioning, and expected performances that will gives crucial directives and information to designer and implementors. This phase of the work resulted in deliverable provisioning and expected performances. This phase resulted in deliverable ETSI TR 103 843 [i.5].

The present document covers the third of the five items listed above and provides the basis for the related ETSI publications listed below:

* ETSI TR 103 839: Scenarios for evaluation of oneM2M deployments [i.1].
* ETSI TS 103 840: Model for oneM2M Performances Evaluation [i.2].
* ETSI TR 103 841: oneM2M Performance Evaluation Tool (Proof of Concept) [i.3] (the present document).
* ETSI TR 103 842: Demonstration of Performance Evaluation and Analysis for oneM2M Planning and Deployment [i.4].
* ETSI TR 103 843: oneM2M deployment guidelines and best practices [i.5].

## 1.2 Scope of the present document.

The present document presents the tools and libraries developed. More precisely:

1. A **profiler tool**, written in python, whose purpose is to generate requests to the CSE open-source implementations, listen the answers, according to different architectures and KPIs, and build a trace file in a given format.
2. An **OMNeT++ library**, written in C++, coupled with the ned proprietary OMNeT++ language, used to specify the case studies graph topologies, as described in [i.1].

The document is structured as follows:

* Clauses 1 to 3 set the scene and provide references as well as definition of terms, symbols and abbreviations, which are used in the present document.
* Clause 4 describes the profiler tool that will be runned in of the open-source oneM2M implementation, its internal structure, including its interactions with different operating systems (e.g. Linux, Windows, OSX), together with a minimal documentation.
* Clause 5 describes the discrete event OMNeT++ simulator and its instantiation of either deployment model applied on oneM2M case studies, as described [i.1,i.2]. This includes the OMNeT++ library and the ned topology specifications and oneM2M protocol interconnection rules.
* Clause 6 put at work the above elements to the case studies defined in [i.1], highlighting the simulation capabilities of the library, according to the KPI synthetized by the profiler. This will allow to scale up our proof of concept.
* Clause 7 provides the conclusions of this work and present some potential improvements and extensions.

# 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.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long-term validity.

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

[i.1] ETSI TR 103 839

[i.2] ETSI TR 103 840

[i.3] ETSI TR 103 841 (present document)

[i.4] ETSI TR 103 842

[i.5] [TS-0001-Functional\_Architecture-V3\_34\_0.ZIP](https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=36557)

[i.6] [TS-0008-CoAP\_Protocol\_Binding-V3\_9\_0.ZIP](https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=36185)

[i.7] [TS-0009-HTTP\_Protocol\_Binding-V3\_9\_0.ZIP](https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=36335)

[i.8] [TS-0010-MQTT\_protocol\_binding-V3\_1\_0.ZIP](https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=32184)

[i.9] [List of oneM2M deployments](https://www.onem2m.org/using-onem2m/list-of-deployments)

[i.10] [OMNeT++ web page](https://omnetpp.org/)

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

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

**Performance evaluation:** refers to the evaluation of temporal, data transfer volumetry, and scalability aspects of a system.

**Real time constraints:** refers to the dynamic constraints to be fulfilled.

**Key Performance Index (KPI):** refers to a list of criteria to be measured on a oneM2M given implementation.

**oneM2M deployment:** refers to the mapping of a IoT applications on a oneM2M infrastructure.

**oneM2M standard implementation:** refers to the list of the implementations of the oneM2M standard.

**Profiler:** refers to a monitoring tool measuring KPIs.

**Platform evaluation tool:** refers to the simulation environment that is used to calculate/demonstrate the performance of the oneM2M standard.

**Single/multiple horizontal/vertical domains:** refers to the interaction capability of many oneM2M infrastructures from different domains.

**Guidelines and good practices:** refers to a methodological document that gives hints to deploy a oneM2M infrastructure.

## 3.2 Symbols

void

## 3.3 Abbreviations

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

ADN Application Dedicated Node

AE Application Entity

API Application Program Interface

CIN Content INstance

CNT Container

CSE Common Service Entity

CRUD Create, Read, Update, Delete

ETSI European Telecommunications Standards Institute

FGT Formal Graph Topology

GGP Guidelines and Good Practices

IN-CSE Infrastructure Node - Common Services Entity

IoT Internet of Things

JSON JavaScript Object Notation

KPI Key Performance Indicators

KCP Key Configuration Parameters

M2M Machine-to-Machine

M2MSP M2M Service Provider

Mca Reference Point for M2M Communication with AE

Mcc Reference Point for M2M Communication with CSE

MM Meta Model

MN-CSE Middle Node - Common Services Entity

OASD oneM2M Application Scenario Descriptor

OCPD oneM2M CSE Performance Descriptor

OIS oneM2M Implementations Standard

OM2M Eclipse OM2M - Open-Source platform for M2M communication

ONO oneM2M Numerosity Objects

OSDD oneM2M Solution Deployment Descriptor

OS Open Source

PE Performance Evaluation

PER Packet Error Rate

PET Platform Evaluation Tool

RT Real-Time

RTR Real Time Requirements

SW Software

TC Technical Committee

TR Technical Report

TS Technical Specification

UCT Use Case Template

# 4 oneM2M profiler

The simulation of an IoT infrastructure in terms of equipment, service layer and application requires calibration of the inputs of the simulator. The different layers of the model proposed in the ETSI TS 103 840 document have an impact on the overall system. The aim here is to propose a measurement architecture making it possible to extract measurements from particular real execution on given equipment, from a particular oneM2M stack for a particular application. Those measurements should be generalized at the simulator level to give KPIs having values as close as possible to the reality of deployment on other equipment for IoT applications with different configuration and behavior.

## 4.1 Structure

## The minimum oneM2M-level configuration needed to perform measurements requires a CSE and a client application making REST requests to the CSE. These two entities can be on the same equipment or on different equipment communicating via a network. Added to this is an entity called profiler responsible for observing the CSE instance and its use of the resources of the hardware equipment. These observations are then processed by the profiler and saved on permanent media in a file.

The implementation of a CSE involves complex mechanisms such as the use of thread, processing priority, recovery, cache, etc. The observation of each of these mechanisms and their impact can only be done by direct insertion of a probe into the code of a CSE itself. This makes the approach not portable and restricted to experts in a specific stack. The choice for profiling was to see the CSE as a black box and to put probes at the operating system level and therefore the resource requests made by the CSE to the operating system of a machine. This approach makes the profiler more independent of CSE implementations but still maintains a dependence on the type of operating system. The precision of the metrics obtained remains sufficient to detect and give the order of magnitude of saturation phenomena (for example of memory) or inadequate response time for a use case.

## 

Profiler

Computer

oneM2M Stack (ACME, OM2M, Mobius, …)

1-request run CSE (ACME, OM2M, Ocean)

3 – request to start to observe computer resources (processor, memory, …)

oneM2M Resources generator

Computer

5 – CRUD request on oneM2M resources to a CSE

4-observe computer resources usage

6 – **request** to stop to observe computer resources (AE create)

7 – save on file computer resources usage for a specific oneM2M resources

2-run CSE (ACME, OM2M, mobius)

8 – request Stop CSE

9 – Stop CSE

*Figure XXX: Interactions between the profiler and the oneM2M system*

## Figure XXX gives an overall view of the system and the exchanges with the different steps of the profiling. In the initial state, the profiler is executed on a machine and an onM2M query generation client is launched, then the experimental protocol is started:

1. The query generator begins by asking the profiler to launch and start monitoring a particular oneM2M CSE.
2. The CSE is executed on the machine
3. The query generator warns the profiler that it must begin its observation on a particular sequence of queries that it will perform on the CSE
4. The profiler launches its observation
5. The query generator executes a set of queries to the CSE
6. The query generator notifies the profiler that the queries on the CSE have been completed and that it can generate the observation statistics of the hardware resources used by the CSE
7. The profiler saves its observations in a file which will be used as input to the simulator
8. When the entire profiling plan is completed the query generator asks the profiler to stop the CSE
9. The profiler stops the CSE and terminates
10. The query generator ends

This simple exchange protocol allows to configure various query generation scenarios. To do this, it is enough to define the oneM2M query plan at the query generator level and on the other hand to explain the method for calculating the statistics on the profiler side for this particular query sequence.

## 4.2 Operating system interaction

In this black box profiling model, there is a strong dependency on the operating system. Those provide libraries for relating the use of hardware and system resources by a particular running process. There are many possible metrics. Here too, the structure of the profiler as an independent entity on the same machine as the CSE makes it possible to enrich the metrics according to needs. For example in the UNIX world and more particularly Linux, all the files under the /proc directory allow, via a set of files, to know the use of resources via measurements directly in the kernel or via counters. For example, the “stat” file gives information on the number of processor ticks used by the process, the “statm” file gives information on the use of RAM and virtual memory, etc.

## 4.3 Profiler usage

4.3.1 Insatallation and configuration

The installation and configuration process involves both the IoT stack you want to observe, the profiler and the query generator. The first step begins with installing the IoT stack you want to observe on the machine concerned. You must configure it (name of the CSE, address, etc.). Please note that version 1 of query generator only accepts the http protocol.

The second step concerns the installation of the profiler and the query generator to be downloaded from the ETSI git. All of these programs are written in python and therefore require the python interpreter to be installed on the CSE machine and the query generator machine.

The third step concerns the profiler. The profiler program must be put in the directory where the CSE code is located. In version 1 of the profiler only CSEs from the ACME, Mobius and OM2M distributions are created. Two parameters must be configured in the profiler: the host variable specifying the IP address of the profiler and port specifying the port which will be used to communicate with the profiler.

The fourth step concerns the configuration of the query generator. The connection with the profiler must be made through the HOST and PORT variables of the generator which must have the same values as the host and port variables of the profiler. Then you must configure the information concerning the IoT stack through the variables CSE\_URL\_XXX and ORIGIN\_XXX (XXX is to be replaced by the correct IoT stack name) which gives respectively the URL of the CSE and the authentication necessary for connection.

Finally, the last step of configuration concerns the connection with an http server in the case of using the notification mechanism via the HTTTP\_SERVEUR variable. For example, the ACME notification server can be used.

4.3.2 Execution

The first step is to run the profiler via the command: “python3 profiler.py”

The second step is to run the query generator. Two modes exist:

* Interactive mode: python3 oneM2M\_Ressources.py manual, in this mode an interaction with the user allows you to choose the IoT stack, the type of resource, the number of operations to be done on the same type of resources in order to generate averages
* Automatic mode: python3 oneM2M\_Ressources.py auto [OM2M/ACME/MOBIUS] [number of iteration per test]", in this mode a pre-defined scenario allows you to create and destroy the most classic oneM2M resources by choosing the stack to execute and the number of requests per resource to generate the statistic.

These two executions generate in the directory where the profiler is located and on its machine a file with the statistics indexed by the name of the stack.

## 4.4 Profiler output format (Thierry)

# 5 oneM2M simulator

## 5.1 OMNeT++ at the glance map

## 5.2 oneM2M deployment model briques du simulateur samir

## 5.3 Mapping on ONMeT++ library and the ned topology samir

- mapping des classes  
 - mapping des descripteur de performances (profiler)

- mapping des # sondes et leur lien avec les KpIs listees dans le D2

## 5.4 Installation and configuration map

# 6 Frameworks @ work

## 6.1 Mapping Scenario Description

6.11.Running example map

Image de la topologie

6.1.2 OMNeT++ format (Ned and ini ) dans le cas de l’exemple (Samir)

6.1.3 Synthetizing KPIs

Instanciation dans le simulateur des valeurs fournies par le profiler

## 6.4 Performance evaluation analysis

# 7 Conclusions, improvement, extensions

Annex A: Source code

See https://labs.etsi.org/rep/iot/......  
See https://labs.etsi.org/rep/iot/.....

Annex B: Change history

| Date | Version | Information about changes |
| --- | --- | --- |
| January 2024 | 0.0.1 | Early draft proposal (Inria) |
| February 2024 | 0.0.2 | Insert profiler information (T. Monteil) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |