ETSI TS 118 119 V2.3.0 (2020-11)

oneM2M Abstract Test Suite and Implementation eXtra Information for Test (oneM2M TS-0019 version 2.3.0 Release 2)

Reference
RTS/oneM2M-000019v2

Keywords
interoperability, M2M, testing

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.

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

Contents

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 Specification (TS) has been produced by ETSI Partnership Project oneM2M (oneM2M).

1 Scope

The present document contains the Abstract Test Suite (ATS) for oneM2M as defined in ETSI TS 118 101 [1] and ETSI TS 118 104 [2] in compliance with the relevant requirements and in accordance with the relevant guidance given in ISO/IEC 9646-7 [5].

The objective of the present document is to provide a basis for conformance tests for oneM2M products giving a high probability of inter-operability between different manufacturers' equipment.

The ISO standard for the methodology of conformance testing (ISO/IEC 9646-1 [3] and ISO/IEC 9646-2 [4]) as well as ETSI TS 118 115 [i.2] are used as a basis for the test methodology.

2 References

2.1 Normative references

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

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

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 necessary for the application of the present document.

[1] ETSI TS 118 101: "oneM2M; Functional Architecture (oneM2M TS-0001)".

[2] ETSI TS 118 104: "oneM2M; Service Layer Core Protocol (oneM2M TS-0004)".

[3] ISO/IEC 9646-1 (1994): "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 1: General concepts".

[4] ISO/IEC 9646-2 (1994): "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 2: Abstract Test Suite specification".

[5] ISO/IEC 9646-7 (1995): "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 7: Implementation Conformance Statements".

[6] ETSI ES 201 873-1 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

[7] oneM2M TS-0018: "Test Suite Structure and Test Purposes".

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 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] ETSI TS 118 115: "oneM2M; Testing Framework (oneM2M TS-0015)".

[i.3] ETSI TS 118 125: "Definition of product profiles (oneM2M TS-0025)".

[i.4] ETSI TS 118 111: "oneM2M; Common Terminology (oneM2M TS-0011)".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ISO/IEC 9646-1 [3], ISO/IEC 9646-7 [5] and ETSI TS 118 115 [i.2] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TS 118 111 [i.4] and the following apply:

AE	Application Entity
APT	Abstract Protocol Tester
ATM	Abstract Test Method
ATS	Abstract Test Suite
CoAP	Constrained Application Protocol
CSE	Common Service Entity
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IUT	Implementation Under Test
IXIT	Implementation eXtra Information for Test
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
MTC	Main Test Component
PA	Platform Adaptor
PICS	Protocol Implementation Conformance Statement
PTC	Paralell Test Component
PX	PiXit
SA	System Adaptor
SUT	System Under Test
TC	Test Case
TCP	Transmission Control Protocol

TP	Test Purposes
TSS	Test Suite Structure
TTCN	Tree and Tabular Combined Notation
UDP	User Datagram Protocol
UT	Upper Tester
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 Abstract Test Method (ATM)

5.1 Abstract protocol tester

An abstract Protocol Tester (APT) is a process that provides behaviours for testing an IUT by emulating a peer IUT at the same layer, and enabling to address a single test objective.

APTs used by the oneM2M test suite are described in figure 5.1-1. The test system will simulate valid and invalid protocol behaviour, and will analyse the reaction of the IUT.

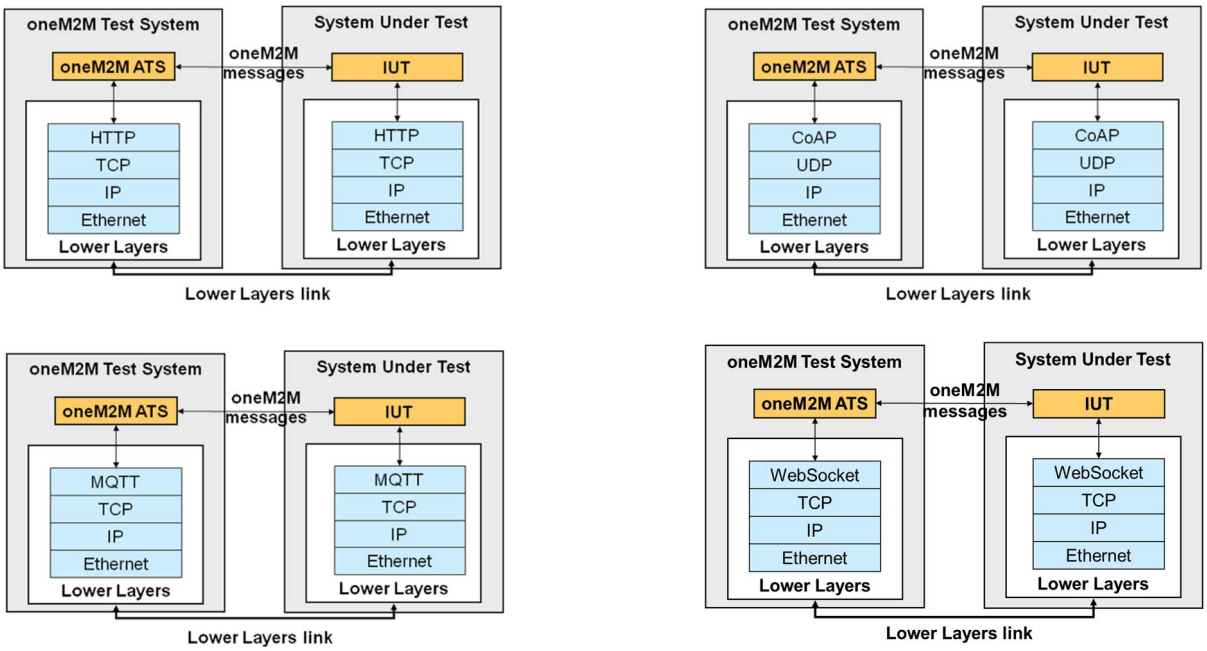


Figure 5.1-1: Abstract protocol testers - oneM2M

As figure 5.1-1 illustrates, the corresponding ATS needs to use lower layers to establish a proper connection to the System Under Test (SUT) over a physical link (Lower layers link). Three different lower layers have been specified corresponding to the binding protocols considered in oneM2M: HTTP, CoAP and MQTT.

5.2 Test Configuration

5.2.1 AE Test Configuration

Test configurations are defined to test different entities such as CSE and AE, etc.

Figure 5.2.1-1 shows a AE test configuration which is mapped to CF03 in clause 6.3.3.3 in ETSI TS 118 115 [i.2] and aligns with conformance test system architecture in clause 6.3.3.2 in ETSI TS 118 115 [i.2].

The TTCN-3 Test Component in Test System sends triggering actions or behaviour to the Upper Tester Application of SUT through upper tester transport link *Ut* while the IUT sends/receives oneM2M service primitives through Mca to/from CSE in Test System.

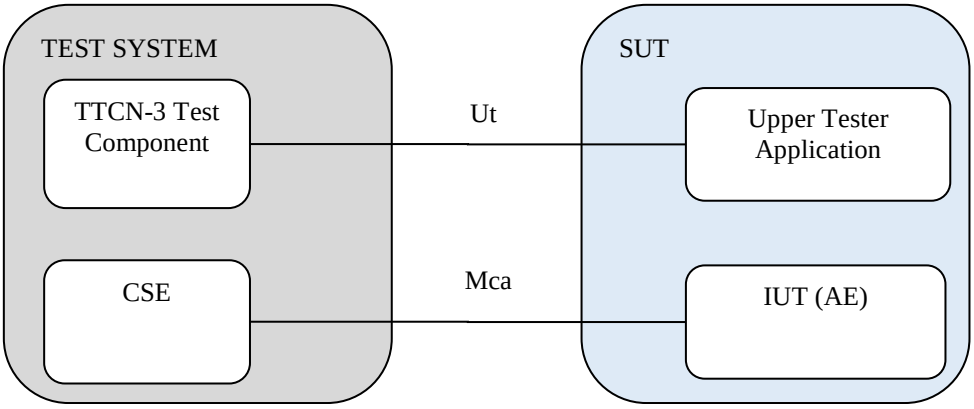
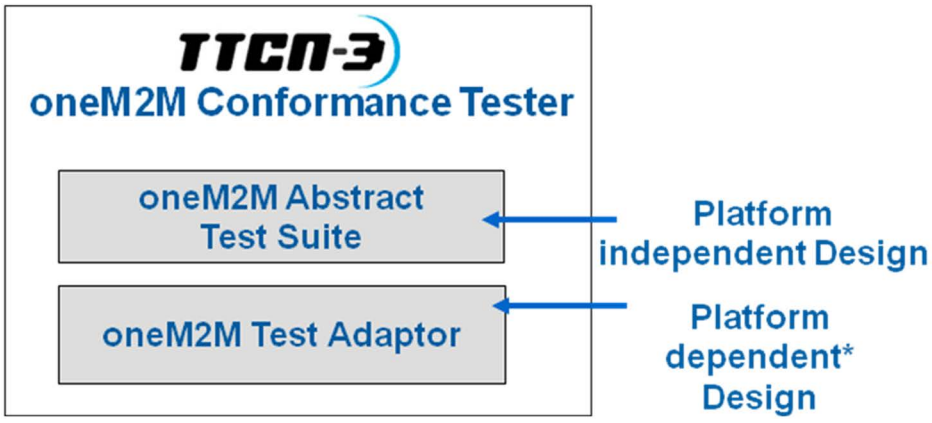


Figure 5.2.1-1: AE test configuration

5.3 Test architecture

The approach for the implementation of an Abstract Protocol Tester selected in oneM2M follows the recommendation of the oneM2M Testing Framework (ETSI TS 118 115 [i.2]) where the TTCN-3 language and its architecture are recommended.

Following this recommendation the oneM2M tester architecture comprises a non-platform dependent Test Suite, and a platform dependent part.



NOTE: However, it can be implemented in a semi-independent manner, which will minimize the dependency to those elements.

Figure 5.3-1: High level oneM2M Test Architecture

- **oneM2M TTCN-3 Abstract Test Suite:** the test suite is platform independent, and it is the cornerstone of the architecture. It allows a complete decoupling between the test suite and the rest of the test system. The test suite is composed of a complete set of test cases covering oneM2M requirements specified by ETSI TS 118 101 [1] and ETSI TS 118 104 [2].

- **oneM2M System Adaptor:** this is the platform dependent part that includes adaptors and codecs (out of the scope of the present document). This part of the architecture definition depends on the specific platform (e.g. Windows® or Linux®) and test tool on which the tester is going to run.

Figure 5.3-2 shows the oneM2M TTCN-3 test architecture design used for the oneM2M ATS. The Test Suite needs to interact with the System Adaptor to implement the collection of TTCN-3 test cases that are intended to be used to test the oneM2M IUTs.

The oneM2M TTCN-3 test cases implement the test algorithms specified in the TSS&TP document (oneM2M TS-0018 [7]), including verdict logic that allows pass/fail diagnosis.

The test algorithms use the interfaces defined in ETSI TS 118 101 [1] and ETSI TS 118 104 [2] (mca, mcc) in order to:

1) control the test event to be sent towards the IUT; and

2) observe the test events received from the IUT.

In TTCN-3 these two interfaces have been implemented through a set of logical TTCN-3 ports (mcaPort and mcaPortIn for mca interface, and mccPort and mccPortIn for mcc interface) which allows oneM2M message primitives exchange with the IUT.

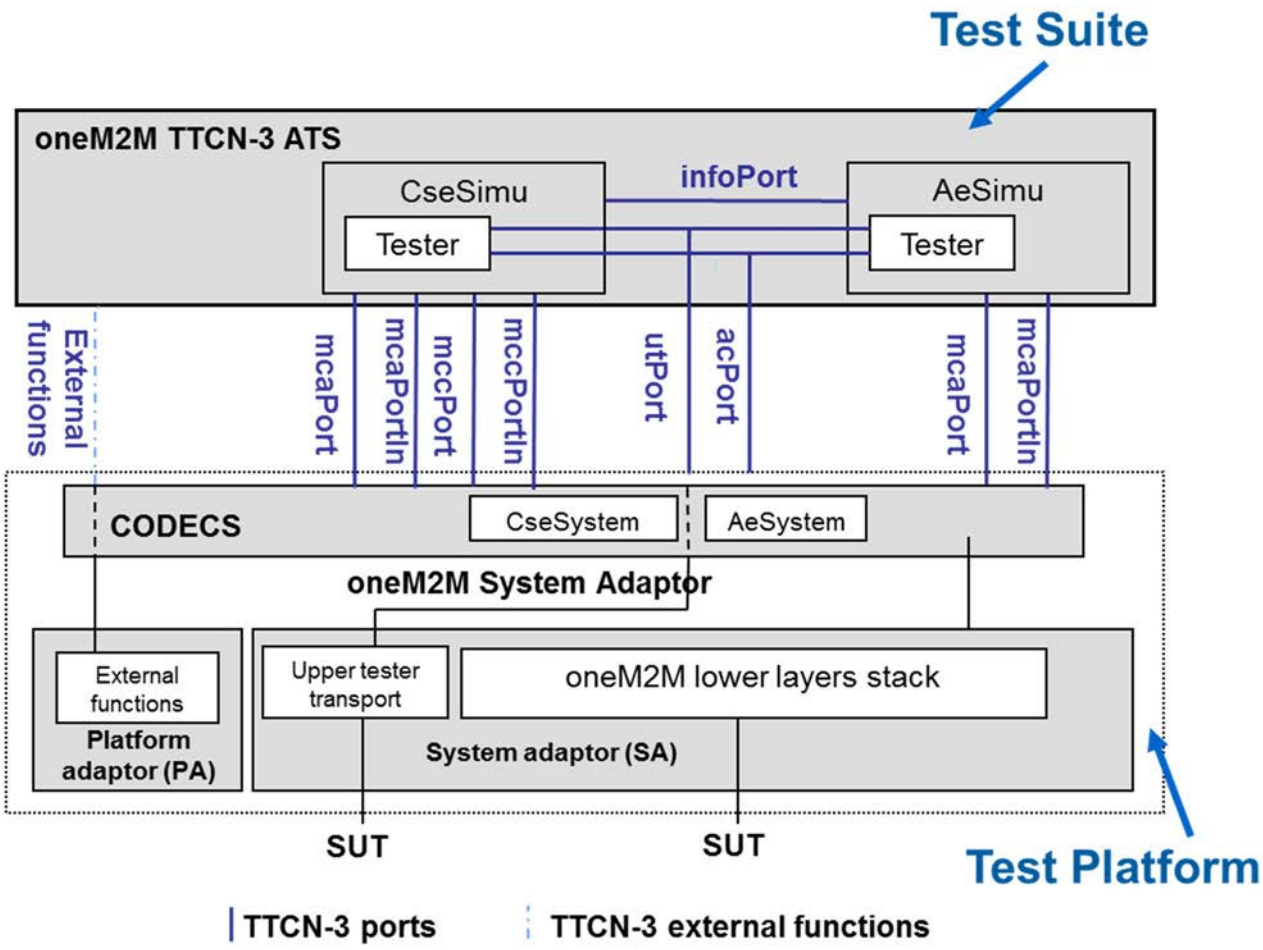


Figure 5.3-2: oneM2M Test Architecture

The oneM2M primitive messages have been mapped into TTCN-3 structure. Through this mapping, the TTCN-3 is able to build and send these messages, as well as receive them via the ports defined above.

Additionally, the test cases are able to control and configure the test platform through a dedicated port called acPort while port utPort enables oneM2M TTCN-3 Test Component module to trigger specific action or behaviour on IUT. TTCN-3 Test Components can also exchange information through a dedicated port called infoPort.

To build up a tester, the test platform needs to be also developed (out of scope). This test platform is composed of three adaptation layers:

- PA (Platform Adaptor) layer functionality implements the communication between the TTCN-3 modules and external elements that constitute the test tool such as timers and external functions. The External functions are a powerful resources supported by TTCN-3 language. An External function is a function declared at the TTCN-3 level but implemented at the native level.
- SA (System Adaptor) layer functionality is divided into two modules:
  - oneM2M lower layers stack module implements the communication with the IUT and carries out the oneM2M primitives messages sent to or received from the IUT. This module is based on TCP or UDP depending on the binding supported by the IUT. The binding is a system adaptor parameter.
  - Upper Tester Transport module implements functions that enable triggering specific actions or behaviour on the IUT.
- CODECS layer is the part of the tester to encode and decode messages between the TTCN-3 abstract internal data representation and the format required by the related base standard which the IUT understands. Several CODECS are required in oneM2M tester to cope with the bindings considered in oneM2M (HTTP, CoAP, MQTT) and the serialization methods (xml, json).

5.4 Ports and ASPs (Abstract Services Primitives)

5.4.0 Introduction

The oneM2M ATS implements the following ports:

- The mcaPort and mcaPortIn.
- The mccPort and mccPortIn.
- The acPort.
- The utPort.
- The InfoPort.

5.4.1 mcaPort, mcaPortIn, mccPort, mccPortIn

These ports are used to send and receive the following message sets:

- Request Primitives messages in accordance with ETSI TS 118 104 [2].
- Response Primitives messages in accordance with ETSI TS 118 104 [2].

Two primitives are currently defined for these ports indicated in table 5.4.1-1:

1) The M2MRequestPrimitive - to send or receive oneM2M messages to/from the IUT. Depending on the IUT to be tested:
   a) If the IUT is an AE, these messages are either received or sent by the tester which is associated with the CSE role through the mcaPortIn or the mcaPort respectively.
   b) If the IUT is a CSE, these messages are either sent or received by the tester when it plays the AE role through the mcaPort or the mcaPortIn respectively, or sent or received by the tester when it plays the CSE role through the mccPort or the mccPortIn respectively.
2) The M2MResponsePrimitive - to send or receive oneM2M messages to/from the IUT. Depending on the IUT to be tested:
   a) If the IUT is an AE, these messages are either sent or received by the tester which is associated with the CSE role through the mcaPortIn or the mcaPort respectively.

b) If the IUT is a CSE, these messages are either sent or received by the tester when it plays the CSE role through the mccPortIn or the mccPort respectively, sent or received by the tester when it plays the AE role through the mcaPortIn or mcaPort respectively.

Both primitives contain another parameters that permits to dynamically configure the test adaptor for every single sending. These parameters are:

- Host: IP address of the IUT.
- XML Namespace.
- Protocol binding.
- Serialization.
- ForceFields: used to force invalid or empty values to certain attributes. This behaviour shall be implemented by the System Adaptor.

Table 5.4.1-1: Mapping of TTCN-3 Primitives to oneM2M Service Primitives

TTCN-3 Primitive	oneM2M Message	Direction	IUT
M2MRequestPrimitive	Request Primitive	← →	AE
	Request Primitive	→ ←	CSE
M2MResponsePrimitive	Response Primitive	→ ←	AE
	Response Primitive	→ ←	CSE

5.4.2 utPort

5.4.2.0 Introduction

The utPort is included in the oneM2M ATS in order to be able to stimulate the IUT and receive extra information from IUT upper layers. For instance, the utPort can be applied to automate AE testing shown in clause 5.4.2.1.

5.4.2.1 Usage for Automated AE Testing

The utPort is in charge of the communication between TTCN-3 Test Component module in Test System and the Upper Tester Application in SUT.

Functionalities that TTCN-3 Test Component module and the Upper Tester Application are required to implement are listed as follows:

- TTCN-3 Test Component is able to configure the Test System and send standardized triggering commands to the SUT (Upper Tester Application).
- Upper Tester Application can process the triggering command messages received from Test System (TTCN-3 Test Component) and stimulates IUT to act following the corresponding triggering command (i.e. sending oneM2M service primitives to Test System through Mca port).

oneM2M service Primitive defined for utPort is listed as follows:

- The UtTrigger primitive is used to trigger upper layer events in IUT (i.e. sending oneM2M service primitives to Test System through Mca port).
- The UtTriggerAck primitive is used by IUT to send acknowledgement back to the Test System.

The Upper Tester Application in SUT can be implemented as an embedded source code. An example for implementation of automated AE test for Registration is shown as figure 5.4.2.1-1.

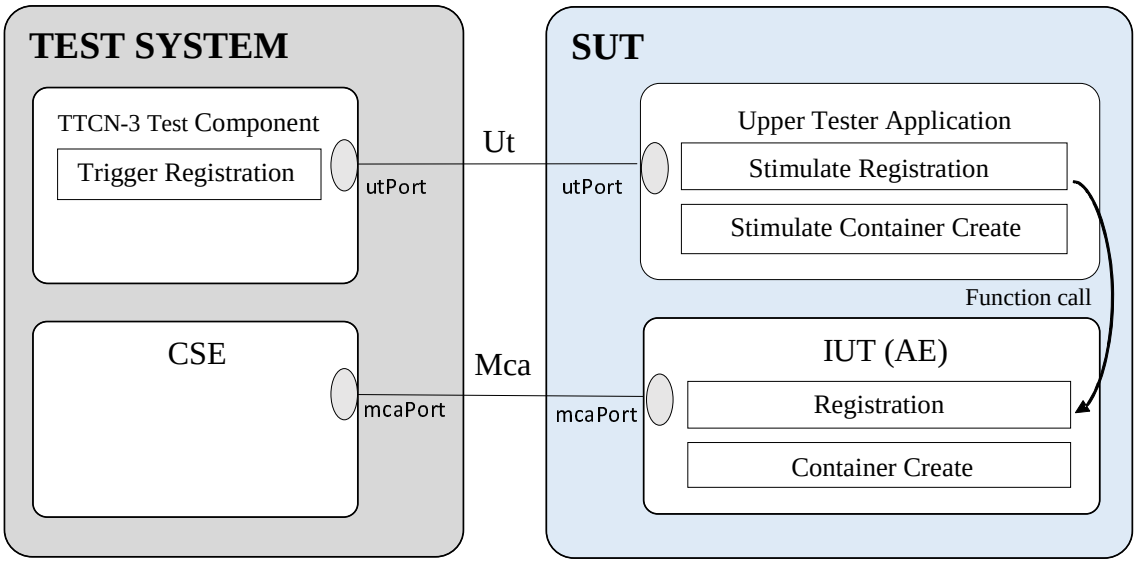


Figure 5.4.2.1-1: Example of automated AE test using Ut interface

5.4.2.2 Upper Tester Control Primitives

5.4.2.2.1 Introduction

The upper tester triggering message is used to transport control commands between Test System and the Upper Tester Application. The control command will contain essential parameters that are required for certain test case.

The upper tester triggering message type maps to particular message formats for exchanging data and those message formats are defined by TTCN-3 primitive as shown in table 5.4.2.2.1-1, *UtTrigger* and *UtTriggerAck* primitive.

Table 5.4.2.2.1-1: Mapping of TTCN-3 Primitives to oneM2M Service Primitives

Upper TesterControl Message Type	TTCN-3 Primitives	Direction	
		TS	UT
Trigger	UtTrigger Primitive	→	
Trigger Acknowledgement	UtTriggerAck Primitive	←	

5.4.2.2.2 UtTrigger and UtTriggerAck Primitives

The UtTrigger primitive is initialized by the Test System to send triggering message to the target IUT as depicted in figure 5.4.2.2.2-1. The IUT will send acknowledgement message back to the Test System using UtTriggerAck primitive if trigger message is successfully transported to the IUT. Then IUT starts interaction with Test System through oneM2M request and response primitives.

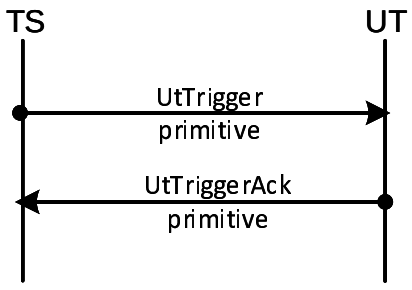


Figure 5.4.2.2.2-1: Trigger message flow

Table 5.4.2.2.2-1 defines UtTrigger and UtTriggerAck primitives including oneM2M data types to which are mapped as well as examples to show how to implement UtTrigger and UtTriggerAck primitives.

Table 5.4.2.2.2-1: UtTrigger and UtTriggerAck Primitive

<table>
<tr><th>Ut Control Primitive</th><th>Mapping to oneM2M data types</th><th>Description</th><th>Reference</th><th>Triggering Message</th><th>HTTP message</th></tr>
<tr><td rowspan="2">UtTrigger Primitive</td><td rowspan="2">request Primitive</td><td rowspan="2">ONLY essential parameters included for certain test case
See note 1</td><td rowspan="2">ETSI TS 118 104 [2]</td><td colspan="2">Example A:
If the test objective is to test "Test System triggers IUT to execute a test case for creation of <AE> with labels attribute under a CSEBase resource", then the triggering message would be serialized as following.</td></tr>
<tr><td>Request
<pre>
{
  "m2m:rqp" :{
      "op": 1, //indicate CREATE
operation
      "ty": 2, //indicate AE
resource type
      "to": {TEST_SYSTEM_ADDRESS},
      "pc": {
        "m2m:ae": {
          "lbl":"UNINITIALIZED"
//indicate that attribute labels
needs to be included
        },
      }
      "rvi": "2a"
  }
}
</pre></td><td>Request
<pre>
POST /{SUT_UT_APPLICATION_URL}
HTTP/1.1
Host: {SUT_IP_ADDRESS:PORT}
Content-Length: {PAYLOAD_LENGTH}
Content-Type: application/json

{
  "m2m:rqp" :{
      "op": 1, //indicate CREATE
operation
      "ty": 2, //indicate AE
resource type
      "to": {TEST_SYSTEM_ADDRESS},
      "pc": {
        "m2m:ae": {
          "lbl":"UNINITIALIZED"
//indicate that attribute labels
needs to be included
        }
      },
      "rvi": "2a"
  }
}
</pre></td></tr>
</table>

<table>
<tr><th>Ut Control Primitive</th><th>Mapping to oneM2M data types</th><th>Description</th><th>Reference</th><th>Triggering Message</th><th>HTTP message</th></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2">Example B:
If the test objective is to test "Test System triggers IUT to execute a test case for delete of a <AE> resource.", then the triggering message would be serialized as following.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>Request
<pre>{
  "m2m:rqp" :{
    "op": 4, //indicate DELETE operation
    "to": {TARGET_AE_RESOURCE_ADDRESS}, //indicate Target AE resource address
    "rvi": "2a"
  }
}</pre></td><td>Request
<pre>POST /{SUT_UT_APPLICATION_URL} HTTP/1.1
Host: {SUT_IP_ADDRESS:PORT}
Content-Length: {PAYLOAD_LENGTH}
Content-Type: application/json

{
  "m2m:rqp" :{
    "op": 4, //indicate DELETE operation
    "to": {TARGET_AE_RESOURCE_ADDRESS}, //indicate Target AE resource address
    "rvi": "2a"
  }
}</pre></td></tr>
<tr><td>UtTriggerAck Primitive</td><td>responsePrimitive</td><td>ONLY responseStatusCode attribute included

See note 2</td><td>ETSI TS 118 104 [2]</td><td>Response
<pre>{
  "m2m:rsp": {
    "rsc": 2000
  }

}</pre>For any triggering response, it only contains a response status code, and the response status code for the triggering operation can only be set to either 2000 (OK) or 4000 (BAD_REQUEST) according to the rules for triggering operations.</td><td>Response
<pre>HTTP/1.1 200 OK
X-M2M-RSC: 2000</pre></td></tr>
<tr><td colspan="6">NOTE 1: Additional rules defined in table 5.4.2.2.2-3 are also applied.
NOTE 2: Attribute response status code is defined in table 5.4.2.2.2-3.</td></tr>
</table>

Table 5.4.2.2.2-2: Rules for defining UtTrigger and UtTriggerAck primitives

① **UtTrigger primitive is represented in requestPrimitive serialized in JSON format.** ② **UtTrigger primitive shall be interpreted as follows:** – Any attribute/parameter containing a value shall be present and equal in the triggered request primitive. – Any attribute/parameter containing "UNINITIALIZED" value shall be present in the triggered request primitive. – Any other attribute/parameter shall comply with ETSI TS 118 104 [2]. ③ **Parameters within UtTrigger are listed as following:** – operation: (**mandatory**)operation type that IUT is triggered to perform. – resourceType: (**optional**)resource type of a target resource against which IUT is triggered to perform certain operation. – to: (**mandatory**)target resource against which IUT is triggered to perform certain operation. – primitiveContent:(**optional**)represents the resource attributes that shall be included in the requestPrimitive.

Table 5.4.2.2.2-3: Definition of ResponseStatusCode for UtTriggerAck primitive

Response Status Code Description	Response Status Code Value	Interpretation
OK	2000	The SUT receives successfully the triggering message from Test System
BAD_REQUEST	4000	The SUT does not interpret correctly the UtTrigger primitive
NOTE: Only above two response status codes are allowed to use in UtTriggerAck primitive.		

5.4.2.2.3 Control Communication Protocol

Protocol used for proceeding communications between TS and Upper Tester Application is designated to the Hypertext Transfer Protocol (HTTP) protocol owning it is an application protocol that is widely supported by most all IoT devices and various intrinsic features such as persistent connection, ease of programming, flexibility, etc.

5.4.2.2.4 Control Message Serialization

Control commands that are wrapped within a request body of HTTP message shall be serialized into JavaScript Object Notation (JSON) because it is very lightweight and easy to parse and generate for machines.

5.4.3 acPort

The acPort is included in the oneM2M ATS in order to be able to control and configure the test adaptor for specific cases.

5.4.4 infoPort

The infoPort is included in the oneM2M ATS in order for the TTCN-3 test components to be able to exchange information such as last response primitives or request primitives received by a component, retrieved primitive contents.

5.5 Test components

5.5.1 Tester

The Tester test component includes a set of ports, timers and variables that are common to the other defined components which are described in table 5.5.1-1.

Table 5.5.1-1: Tester component elements

Name	Instance type	Element type	Description
acPort	port	AdapterControlPort	Port that communicates with the adapter for sending configuration parameters
infoPort	port	InfoPort	Port between test components for exchanging information
utPort	port	UpperTesterPort	Port that communicates with the UT Application for triggering actions on the IUT
tc_ac	timer	N/A	Timer for the reception of a message
tc_wait	timer	N/A	Timer for the reaction of the IUT to an upper tester primitive
vc_config	variable	Configurations	Configuration being used for the given test case
vc_testSystemRole	variable	TestSystemRole	Role of the test component
vc_resourcesList	variable	MyResourcesList	List of all resources created by the test system on the IUT
vc_resourcesIndexToBeDeleted	variable	IntegerList	List of indexes of resources created by the test system on the IUT that need to be deleted
vc_acpAuxIndex	variable	integer	
vc_request	variable	MsgIn	Latest request primitive received/sent
vc_response	variable	MsgIn	Latest response primitive received/sent
vc_aeSimu	variable	default	Reference to the default behaviour for an AeSimu component
vc_cseSimu	variable	default	Reference to the default behaviour for an CseSimu component
vc_primitiveContentRetrievedResource	variable	PrimitiveContent	Latest content of a RETRIEVE operation
vc_myInterfaces	variable	Interfaces	Parameters for the ports of the given component: • Port (mcaPort, mcaPortIn, mccPort, mccPortIn) • Host (SUT IP address :port) • Protocol binding • Serialization

Note that vc_aeSimu and vc_cseSimu are not common to the other defined test components, but those variables are required in Tester for the correct activation/deactivation of default behaviours.

5.5.2 AeSimu

The AeSimu test component extends the Tester component by adding elements specific to an AE entity. Table 5.5.2-1 summarizes those elements.

Table 5.5.2-1: AeSimu component elements

Name	Instance type	Element type	Description
mcaPort	port	oneM2MPort	Port that implements the mca interface when test system is the client (sending requests)
mcaPortIn	port	oneM2MPort	Port that implements the mca interface when test system is the server (receiving requests)
vc_ae2	test component	AeSimu	Reference to the AE2 component when required
vc_cse1	test component	CseSimu	Reference to the CSE1 component when CF02 is used
vc_auxiliaryAe2Up	variable	boolean	Flag to indicate that AE2 component has been started
vc_aeAuxIndex	variable	integer	Index of the AE resource in vc_resourcesList

5.5.3 CseSimu

The CseSimu test component extends the Tester component by adding elements specific to an CSE entity. Table 5.5.3-1 summarizes those elements:

Table 5.5.3-1

Name	Instance type	Element type	Description
mcaPort	port	OneM2MPort	Port that implements the mca interface when test system is the client (sending requests)
mcaPortIn	port	OneM2MPort	Port that implements the mca interface when test system is the server (receiving requests)
mccPort	port	OneM2MPort	Port that implements the mcc interface when test system is the client (sending requests)
mccPortIn	port	OneM2MPort	Port that implements the mcc interface when test system is the server (receiving requests)
vc_ae1	test component	CseSimu	Reference to the CSE1 component when CF02 (CseSimu as master) is used
vc_localResourcesList	variable	MyResourcesList	List of all resources created by the IUT on the test system
vc_localRemoteCseIndex	variable	integer	Index of the remoteCSE resource in vc_localResourcesList representing the IUT (CSE)
vc_remoteCseIndex	variable	integer	Index of the remoteCSE resource in vc_resourcesList representing the test system
vc_cSEBaseIndex	variable	integer	Index of the CSEBase resource in vc_localResourcesList of the test system
vc_cseType	variable	CseTypeID	CSE type of the test system (default is MN)

5.6 Test strategy

This clause introduces the test strategy being used for the TTCN-3 test cases. The chosen strategy permits to have a clear structure of the code that facilitates an easy navigation throw the different test steps.

The use of the TTCN-3 MTC and PTC(s) is as depicted in figure 5.6-1.

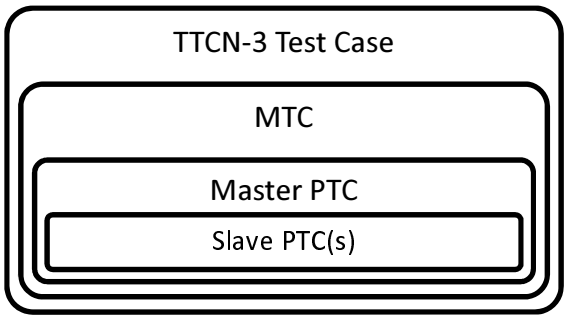


Figure 5.6-1

At the start of the test case execution, the MTC is created. Then, the MTC executes the following steps:

Step 1) initialization of the master PTC.

Step 2) initialization of some parameters if required for the permutation test cases.

Step 3) running of the appropriate function on the master PTC. The function run on the master PTC implements a given Test Purpose. Such function follows a code structure as indicated here below:

- Local Variables, declaration of local variables.
- Test Control, checking IUT capability parameters required for the proper execution of the test.
- Test Component Configuration, that initializes the given test component and other test components acting as slave PTC(s) as required by a given configuration.

- Test adapter configuration, that configures the test adapter throw the acPort if required.
- Preamble, that implements the necessary test steps as described in the Initial conditions of a Test Purpose. It may also implement additional test steps which are required for the correct execution of the test.
- Test body, that implements the test steps as described in the Expected behaviour of a Test Purpose.
- Postamble, that implements the necessary test steps to bring the IUT back to the initial state.
- Tear down, that finalizes properly the TTCN-3 ports used by the different test components depending on the configuration.

While master PTC follows the test structure described above, slave PTC(s) run only certain procedures, usually one by one, as mandated by the master PTC.

A procedure usually implements a oneM2M request-response exchange between a given PTC and the IUT, although it can implement any other specific action (sending or reception of a message, several request-response exchanges, etc.).

Step 4) checking of some parameters if required for the permutation test cases.

This test strategy may slightly vary for certain cases where specific requirements need to be fulfilled.

6 Untestable Test Purposes

Void.

7 ATS Conventions

7.0 Introduction

The ATS conventions are intended to give a better understanding of the ATS but they also describe the conventions made for the development of the ATS. These conventions shall be considered during any later maintenance or further development of the ATS.

The ATS conventions contain two clauses, the naming conventions and the implementation conventions. The naming conventions describe the structure of the naming of all ATS elements. The implementation conventions describe the functional structure of the ATS.

To define the ATS, the guidelines of ETSI TS 118 115 [i.2] were considered.

7.1 Testing conventions

7.1.1 Testing states

7.1.1.1 Initial state

All test cases start with the function f_preamble_XYZ. This function brings the IUT in an "initialized" state by performing some actions such as registration of AE, creation of auxiliary access control policy resource, creation of additional needed resources.

7.1.1.2 Final state

All test cases end with the function f_postamble_XYZ. This function brings the IUT back in an "idle" state which means deletion of all created resources being used by the test case so that next test case execution is not disturbed.

As necessary, further actions may be included in the f_postamble functions.

7.2 Naming conventions

7.2.1 General guidelines

This test suite follows the naming convention guidelines provided in ETSI TS 118 115 [i.2].

The naming convention is based on the following underlying principles:

- in most cases, identifiers should be prefixed with a short alphabetic string (specified in table 7.2.1-1) indicating the type of TTCN-3 element it represents;
- suffixes should not be used except in those specific cases identified in table 7.2.1-1;
- prefixes and suffixes should be separated from the body of the identifier with an underscore ("_");

EXAMPLE 1: `c_sixteen, t_wait`.

- only module names, data type names and module parameters should begin with an upper-case letter. All other names (i.e. the part of the identifier following the prefix) should begin with a lower-case letter;
- the start of second and subsequent words in an identifier should be indicated by capitalizing the first character. Underscores should not be used for this purpose.

EXAMPLE 2: `f_initialState`.

Table 7.2.1-1 specifies the naming guidelines for each element of the TTCN-3 language indicating the recommended prefix, suffixes (if any) and capitalization.

Table7.2.1-1: TTCN-3 generic naming conventions

Language element	Naming convention	Prefix	Example identifier
Module	Use upper-case initial letter	none	OneM2M_Templates
Group within a module	Use lower-case initial letter	none	messageGroup
Data type	Use upper-case initial letter	none	SetupContents
Message template	Use lower-case initial letter	m_	m_setupInit
Message template with wildcard or matching expression	Use lower-case initial letters	mw_	mw_anyUserReply
Signature template	Use lower-case initial letter	s_	s_callSignature
Port instance	Use lower-case initial letter	none	signallingPort
Test component instance	Use lower-case initial letter	none	userTerminal
Constant	Use lower-case initial letter	c_	c_maxRetransmission
Constant (defined within component type)	Use lower-case initial letter	cc_	cc_minDuration
External constant	Use lower-case initial letter	cx_	cx_macId
Function	Use lower-case initial letter	f_	f_authentication()
External function	Use lower-case initial letter	fx_	fx_calculateLength()
Altstep (incl. Default)	Use lower-case initial letter	a_	a_receiveSetup()
Test case	Use ETSI numbering	TC_	TC_COR_0009_47_ND
Variable (local)	Use lower-case initial letter	v_	v_macId
Variable (defined within a component type)	Use lower-case initial letters	vc_	vc_systemName
Timer (local)	Use lower-case initial letter	t_	t_wait
Timer (defined within a component)	Use lower-case initial letters	tc_	tc_authMin
Module parameters for PICS	Use all upper case letters	PICS_	PICS_DOOROPEN
Module parameters for other parameters	Use all upper case letters	PX_	PX_TESTER_STATION_ID
Formal Parameters	Use lower-case initial letter	p_	p_macId
Enumerated Values	Use lower-case initial letter	e_	e_syncOk

7.2.2 oneM2M specific TTCN-3 naming conventions

Next to such general naming conventions, table 7.2.2-1 shows specific naming conventions that apply to the oneM2M TTCN-3 ATS.

Table 7.2.2-1: oneM2M specific TTCN-3 naming conventions

Language element	Naming convention	Prefix	Example identifier
oneM2M Module	Use upper-case initial letter	OneM2M_	OneM2M_Testcases_
Module containing oneM2M types	Use upper-case initial letter	OneM2M_Types	OneM2M_Types
Module containing types and values	Use upper-case initial letter	OneM2M_TypesAndValues	OneM2M_TypesAndValues
Module containing Templates	Use upper-case initial letter	OneM2M_Templates	OneM2M_Templates
Module containing test cases	Use upper-case initial letter	OneM2M_Testcases	OneM2M_Testcases
Module containing functions	Use upper-case initial letter	OneM2M_Functions	OneM2M_Functions
Module containing external functions	Use upper-case initial letter	OneM2M_ExternalFunctions	OneM2M_ExternalFunctions
Module containing components, ports and message definitions	Use upper-case initial letter	OneM2M_TestSystem	OneM2M_TestSystem
Module containing module parameters	Use upper-case initial letter	OneM2M_Pixits	OneM2M_Pixits

7.2.3 Usage of Log statements

All TTCN-3 log statements use the following format using the same order:

- The TTCN-3 test case or function identifier in which the log statement is defined.
- One of the categories of log: INFO, WARNING, ERROR, TIMEOUT, NONE.
- Free text.

EXAMPLE 1:
```
log("f_utInitializeIut: INFO: IUT initialized");
```

Furthermore, the following rules are applied too:

- All TTCN-3 setverdict statements are combined (as defined in TTCN-3 - ETSI ES 201 873-1 [6]) with a log statement following the same above rules (see example 2).

EXAMPLE 2:
```
setverdict(pass, "TC_ONEM2M_CSE_DMR_CRE_001: Received correct
message");
```

7.2.4 Test Case (TC) identifier

Table 7.2.4-1: TC naming convention

Identifier:	TC_<root>_<gr>_<sgr>_<nn>_<per>		
	<root> = root	ONEM2M	oneM2M
	<gr> = group	CSE	CSE testing
		AE	AE testing
	<sgr> = subgroup	REG	Registration
		DMR	Data Management and Repository
		SUB	Subscription and Notification
		GMG	Group Management
		DIS	Discovery
		LOC	Location
		DMG	Device Management
		CMDH	Communication Management and Delivery Handling
		SEC	Security
	<nn> = sequential number		001 to 999
	<per> = permutation	P1_P2_..PN	Permutation parameters

EXAMPLE: TP identifier: TP/oneM2M/CSE/DMR/CRE/001
TC identifier: TC_ONEM2M_CSE_DMR_CRE_001

7.3 IXIT

The following parameters are used by the oneM2M ATS for the correct execution of the test cases.

Table 7.3-1: oneM2M ATS IXITs

GROUP	IXIT NAME	DESCRIPTION	DEFAULT VALUE
IutParameters	PX_IN_CSE	MN-CSE	true
	PX_MN_CSE	IN-CSE	false
	PX_ASN_CSE	ASN-CSE	false
	PX_SUT_ADDRESS	SUT address	"127.0.0.1:8080"
	PX_UT_IMPLEMENTED	Upper Tester implemented	false
	PX_CSE_NAME	IUT CSE Name	"cseName"
	PX_CSE_ID	IUT CSE-ID with SP-relative-CSE-ID format (relative) according to ETSI TS 118 101 [1], table 7.2-1	"/cseId"
	PX_CSE_RESOURCE_ID	IUT CSE resource ID with Unstructured-CSE-relative-Resource-ID (relative) format according to ETSI TS 118 101 [1], table 7.2-1	"cseResourceId"
	PX_SP_ID	IUT M2M-SP-ID with M2M-SP-ID format (absolute) according to ETSI TS 118 101 [1], table 7.2-1 Unstructured-CSE-relative -Resource-ID	"//om2m.org"
	PX_SUPER_AE_ID	AE-ID with privileges to CREATE at the IUT CSEBase with AE-ID-Stem format (relative) according to ETSI TS 118 101 [1], table 7.2-1	"admin:admin"

GROUP	IXIT NAME	DESCRIPTION	DEFAULT VALUE
	PX_SUPER_CSE_ID	CSE-ID with privileges to CREATE at the IUT CSEBase with SP-relative-CSE-ID format (relative) according to ETSI TS 118 101 [1], table 7.2-1	"/admin:admin"
	PX_ALLOWED_C_AE_IDS		{"C-AllowedAeId"}
	PX_NOT_ALLOWED_C_AE_IDS		{"C-NotAllowedAeId"}
	PX_ALLOWED_S_AE_IDS		{"S-AllowedAeId"}
	PX_NOT_ALLOWED_S_AE_IDS		{"S-NotAllowedAeId"}
	PX_NOT_ALLOWED_APP_ID		"NotAllowedAppId"
	PX_ADDRESSING_METHOD	Addressing method	e_hierarchical
	PX_PRIMITIVE_SCOPE	Primitive scope	e_cseRelative
	PX_WS_PROTOCOL	WebSocket protocol	"oneM2M.R2.0.xml"
	PX_REQUEST_URI	WebSocket context	"/"
	PX_HOSTING_CSE_ID	Hosting CSE-ID for MQTT	"CSE-ID"
	PX_CREDENTIAL_ID	Credential-ID for MQTT	"admin:admin"
	PX_XML_NAMESPACE	XML Namespace	"m2m=""http://www.onem2m.org/xml/protocols"""
	PX_ACOR	AccessControlOriginators	{"all"}
	PX_TCONFIG_IUT	Time to configure IUT after a requested action	10.0

TesterParameters	PX_TS_AE1	AE1 component settings	aeIdStem = ""
			appId = "NMyApp1Id"
			mcaPort and mcaPortIn settings which include per port the following info:
			- Binding:
			  - bindingProtocol
			  - bindingDesc:
			  - tsAddress
			  - localPort
			  - sutAddress
			  - remotePort
			- Serialization

	PX_TS_AE2	AE2 component settings	aeIdStem = ""
			appId = "NMyApp2Id"
			mcaPort and mcaPortIn settings which include per port the following info:
			- Binding:
			  - bindingProtocol
			  - bindingDesc:
			  - tsAddress
			  - localPort
			  - sutAddress
			  - remotePort
			- Serialization

GROUP	IXIT NAME	DESCRIPTION	DEFAULT VALUE
	PX_TS_CSE1	CSE1 component settings	cseName = "CSE1_NAME" cseId = "/CSE1_ID" cseResourceId = "CSE1_RESOURCE_ID" spId = "//onem2m.org" supportedResourceType = {int1, int2, int3, int16} mcaPort, mcaPortIn, mccPort and mccPortIn settings which include per port the following info: - Binding: - bindingProtocol - bindingDesc: - tsAddress - localPort - sutAddress - remotePort - Serialization
	PX_TS_UT	UpperTester settings	url = "http://127.0.0.1:43000/"
ExecutionParameters	PX_RESOURCES_TO_BE_DELETED	(For debugging purposes)	{"MyAe", "MyAccessControlPolicyResource", "SubscriptionVerificationAcp", "MyAcp", "MyRemoteCSEResource"}
	PX_RUN_POSTAMBLE	(For debugging purposes)	true

8 TTCN-3 Verifications

The principles for Verifying the TTCN-3 test code are given in ETSI TS 118 115 [i.2].

All test cases provided with the present document in annex A which correspond to at least one of the product profiles defined in ETSI TS 118 125 [i.3] have been verified at the time of publication of the present document which corresponds with the TTCN-3 code gitlab tag provided in annex A.

Annex A (normative): TTCN-3 library modules

A.1 Electronic annex, zip file with TTCN-3 code

This ATS has been produced using the Testing and Test Control Notation (TTCN) according to ETSI ES 201 873-1 [6].

This test suite has been compiled error-free using two different commercial TTCN-3 compilers.

The TTCN-3 library modules, which form parts of the present document, are contained in the following gitLab tag:

- https://git.onem2m.org/TST/ATS/tags/TDE-2019-0106-TS-0019-baseline-v2_3_0.

Annex B (informative): Bibliography

- ISO/IEC 9646-6 (1994): "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 6: Protocol profile test specification".
- ETSI TS 118 117: "oneM2M Implementation Conformance Statements (oneM2M TS-0017)".
- oneM2M TS-0031: "Feature catalogue".

History

Document history		
V2.3.0	November 2020	Publication